jamk.fi

HYBRIDISOVELLUSTEN KEHITYS MOBIILILAITTEILLE

Satu Tyrväinen

Opinnäytetyö
Toukokuu 2014

Mediatekniikan koulutusohjelma
Tekniikan ja liikenteen ala

JYVÄSKYLÄN AMMATTIKORKEAKOULU
JAMK UNIVERSITY OF APPLIED SCIENCES

JYVÄSKYLÄN AMMATTIKORKEAKOULU
JAMK UNIVERSITY OF APPLIED SCIENCES

KUVAILULEHTI

Tekijä(t) TYRVÄINEN, Satu	Julkaisun laji Opinnäytetyö	Päivämäärä 09.05.2014
	Sivumäärä 62	Julkaisun kieli Suomi
		Verkkojulkaisulupa myönnetty (X)
Työn nimi HYBRIDISOVELLUSTEN KEHITYS MOBIILILAITTEILLE		
Koulutusohjelma Mediatekniikan koulutusohjelma		
Työn ohjaaja(t) MANNINEN, Pasi		
Toimeksiantaja(t) SINISALO, Natanael – MEOM Oy		

Tiivistelmä

Opinnäytetyössä tutkittiin alustariippumatonta hybridisovelluskehitystä. Työssä keskityttiin selvittämään PhoneGap-sovelluskehyksen soveltumista hybridisovelluskehitykseen sekä tutkittiin hybridisovellusten mahdollisuuksia, vahvuuksia ja heikkouksia verrattuna web- ja natiivisovelluksiin.

Opinnäytetyön toimeksiantajana toimi jyväskyläläinen Mainostoimisto MEOM Oy, jolle toteutettiin Aarrejahti-sovellus PhoneGapilla. Sovelluksessa on erilaisia kampanjoita, jotka sisältävät yhden tai useamman aarrejahtikilpailun. Nämä aarrejahdit jaetaan kolmeen eri vaiheeseen: ensimmäisessä ja toisessa vaiheessa käyttäjälle näytetään aarteen sijainnista eri vihjekuvat ja kerrotaan, läheneekö vai loittoneeko kilpailija aarteen sijainnista. Kolmannessa vaiheessa aarteenetsijä tarkistuttaa sovelluksessa aarteen sijainnista löytyvän koodin, jonka ollessa oikein aarteenetsijä täyttä sovellukseen yhteystietonsa palkinnon lähettämistä varten. Sovellus käyttää laitteen sijaintia paikallistaakseen kilpailijan.

PhoneGapin lisäksi opinnäytetyössä vertailtiin kahta käyttöliittymäkehystä: Sencha Touchia ja jQuery Mobilea, joista jälkimmäinen valittiin Aarrejahti-sovelluksen toteutukseen. Opinnäytetyössä pyrittiin tutkimaan tapoja toteuttaa mobiilisovelluksia mahdollisimman vaivattomasti eri päätelaitteille niin, että samaa koodia voitaisiin käyttää kaikilla mobiilialustoilla. Lopuksi tarkasteltiin hybridisovellusten mahdollisuuksia sekä tulevaisuudennäkymiä.

Avainsanat (asiasanat)

Mobiilisovellus, hybridisovelluskehitys, PhoneGap

Muut tiedot

JYVÄSKYLÄN AMMATTIKORKEAKOULU
JAMK UNIVERSITY OF APPLIED SCIENCES

DESCRIPTION

Author(s)	Type of publication	Date
TYRVÄINEN, Satu	Bachelor's Thesis	09.05.2014
	Pages 62	Language Finnish
		Permission for web publication (X)

Title
HYBRID APPLICATION DEVELOPMENT FOR MOBILE DEVICES

Degree Programme
Media Engineering

Tutor(s)
MANNINEN, Pasi

Assigned by
SINISALO, Natanael – MEOM Oy

Abstract

This Bachelor's Thesis studies platform-independent hybrid application development. The study focused on researching PhoneGap application framework's suitability for hybrid application development, and studying the possibilities, strengths and weaknesses of hybrid applications compared to web and native applications.

This thesis was commissioned by MEOM Oy, an advertising agency from Jyväskylä. A Treasure Hunt application was developed for this thesis. The application has a variety of campaigns, which contain one or more treasure hunt contests. The treasure hunts were divided into three stages: in the first and second stages the user is presented with different hint images of the treasure's location and the application tells the user whether he or she is moving closer or farther away from the treasure's location. In the third stage the user types in the treasure's unique code and if the code is correct the application asks the user to fill in contact information for sending his or her prize. The application uses GPS to locate the user's current location.

In addition to PhoneGap, two user interface frameworks were compared in this thesis: Sencha Touch and jQuery Mobile, the latter of which was selected in the implementation of the Treasure Hunt application. This thesis aimed at researching ways to implement mobile applications to various devices effortlessly, that the same code could be used on all mobile platforms. Lastly the potential and future prospects of hybrid applications were analyzed.

Keywords
Mobile application, hybrid application development, PhoneGap

Miscellaneous

Sisältö

Kuviot

Taulukot

1 Opinnäytetyön lähtökohdat

1.1 Toimeksiantaja

Opinnäytetyön toimeksiantajana toimi jyväskyläläinen mainostoimisto MEOM Oy. MEOM Oy tuottaa pääasiallisesti verkkosivustoja Wordpress-julkaisujärjestelmälle, mutta heidän tuotteisiinsa lukeutuvat myös erilaiset painotuotteet, yritysilmeet sekä sisällöntuotto. Verkkosivuissaan MEOM Oy seuraa responsiivisen web-kehityksen periaatteita, mikä mahdollistaa sivustojen selaamisen eri päätelaitteilla, kuten älypuhelimella, tabletilla tai tietokoneella.

1.2 Tehtävän kuvaus ja tavoitteet

Opinnäytetyön tavoitteena oli tutkia mobiililaitteiden alustariippumatonta sovelluskehitystä, ja työssä keskityttiin tarkastelemaan eri lähestymistapoja hybridisovellusten toteuttamiseen mobiilialustoille. Tutkimuksen tuloksena toteutettiin aarteenetsintäsovellus, joka hyödyntää laitteen natiiveja ominaisuuksia, mutta voidaan jakaa eri käyttöjärjestelmille mahdollisimman vaivattomasti.

Usealla toimeksiantajan MEOM Oy:n asiakkaalla on ollut erilaisia kampanjoita, joissa sosiaalisessa mediassa jaetun vihjekuvan perusteella kampanjan osallistujat etsivät Jyväskylän alueelta ”aarteen” voittaakseen kampanjan palkinnon. Osallistujien löytämät aarteet ovat eräänlaisia koodeja, jotka syötetään kampanjan web-sivuilla olevaan lomakkeeseen yhteystietojen kanssa. Ensimmäisenä oikean koodin lähettänyt voittaa kampanjan palkinnon. Näitä aarteenetsintäkampanjoita on tarkoitus tukea opinnäytetyössä toteutettavalla sovelluksella. Sovellus toimii aarteenetsijän apuna; se paikallistaa osallistujan ja kertoo tälle reaaliaikaisesti aarteenetsinnän etenemisestä. Tämän lisäksi osallistujan ei tarvitse avata erillistä web-sivustoa koodin syöttämiseen vaan koodi tarkastetaan jo itse sovelluksessa.

2 Mobiililaitteet

2.1 Mitä ovat mobiililaitteet?

Mobiililaitteiksi lasketaan kaikki ne laitteet, joilla voidaan sekä lähettää että vastaanottaa tietoa ajasta ja paikasta riippumatta. Mobiililaite yhdistetään verkkoon langattomasti yleensä joko wi-fi- tai puhelinverkkoyhteyden kautta. Yleisimpiä mobiililaitteita nykyisin ovat älypuhelimet ja taulutietokoneet eli tabletit.

Sekä älypuhelimet että tabletit ovat yleistyneet maailmalla räjähdysmäisesti viime vuosien aikana. BI Intelligencen vuoden 2013 joulukuussa teettämän tutkimuksen mukaan joka viidennellä ihmisellä on käytössään älypuhelin ja joka 17. omistaa tabletin. Suomessa älypuhelimia on jo 61 prosentilla väestöstä (Arki muuttuu yhä mobiilikeskeisemmäksi 2013). Tablettien myynti Suomessa on kasvanut hurjaa vauhtia; vuoden 2013 tammi-syyskuussa myytiin yli 393 000 tablettia ja kasvua edelliseen vuoteen oli jopa 145 prosenttia (Kodintekniikkaindeksi 2013).

2.2 Käyttöjärjestelmät

2.2.1 Yleistä

Käyttöjärjestelmä on laitteen, kuten tietokoneen tai älypuhelimen, perusohjelmisto, joka ohjaa ja hallinnoi laitteen toimintaa. Tällä hetkellä maailmalla ylivoimaisesti suosituin mobiililaitteiden käyttöjärjestelmä on Googlen kehittämä Android, toisena seuraa iOS ja kolmantena BlackBerryn vuonna 2013 ohittanut Windows Phone (ks. taulukko 1, s. 7).

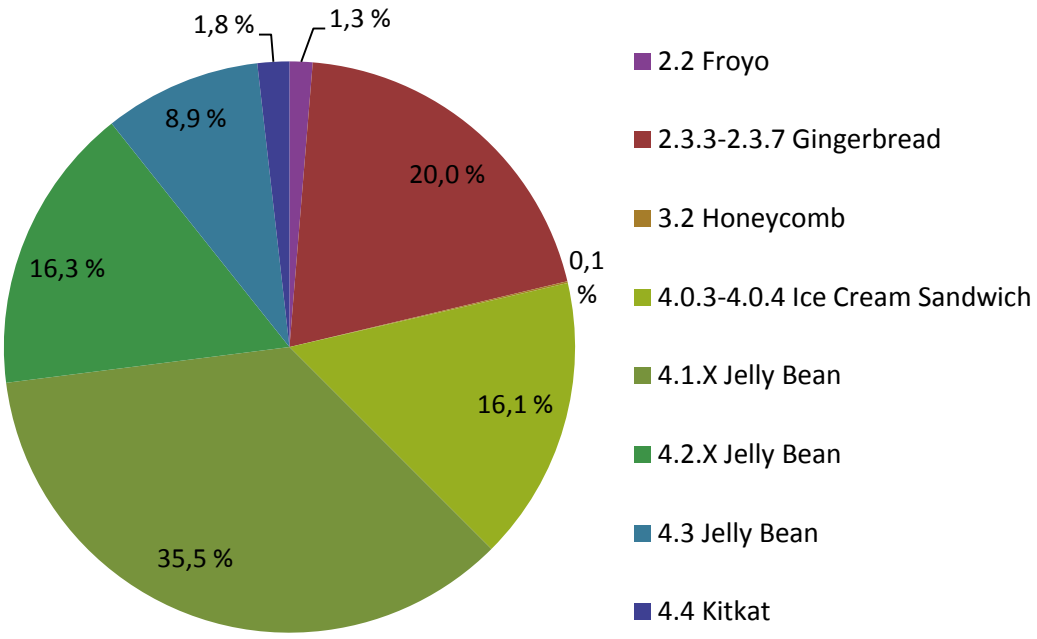
Taulukko 1. Mobiilikäyttöjärjestelmien toimitusmäärät ja markkinaosuudet kolmannella vuosineljänneksellä vuosina 2012 ja 2013 (toimitusmäärät miljoonia) (Android Pushes Past 80% Market Share -- 2013).

Käyttöjärjestelmä	3Q12 toimitusmäärä	3Q12 markkinaosuus	3Q13 toimitusmäärä	3Q13 markkinaosuus
Android	139,9	74,9 %	211,6	81,0 %
iOS	26,9	14,4 %	33,8	12,9 %
Windows Phone	3,7	2,0 %	9,5	3,6 %
BlackBerry	7,7	4,1 %	4,5	1,7 %
Muut	8,4	4,5 %	1,7	0,6 %

2.2.2 Android

Android on Googlen julkaisema mobiililaitteille suunnattu käyttöjärjestelmä, joka perustuu avoimeen lähdekoodiin. Android on maailmalla johtavin mobiilikäyttöjärjestelmä 81 prosentin markkinaosuudella (Android Pushes Past 80% Market Share -- 2013). Sen ensimmäinen versio julkaistiin syyskuussa 2008 HTC:n Dream-älypuhelimelle ja se kantoi nimeä Astro. Android-mobiililaitteilla on useita valmistajia, joista tunnetuimpiin kuuluvat Samsung, HTC, LG ja Sony (Hynninen 2013).

Androidin uusin versio 4.4 Kitkat julkaistiin 31. lokakuuta 2013, mutta sitä käyttää tämän opinnäytetyön tekohetkellä vain 1,8 prosenttia Android-laitteiden omistajista, sillä kaikki vanhemmat mobiililaitteet eivät tue uusimpia Android-versioita. Suosituin Android-käyttöjärjestelmän versio on on 4.1.X Jelly Bean 35,5 prosentin osuudella (ks. kuvio 1, s. 8).



Kuvio 1. Eri Android-käyttöjärjestelmän versioiden suhteelliset osuudet (Platform Versions 2014)

2.2.3 iOS

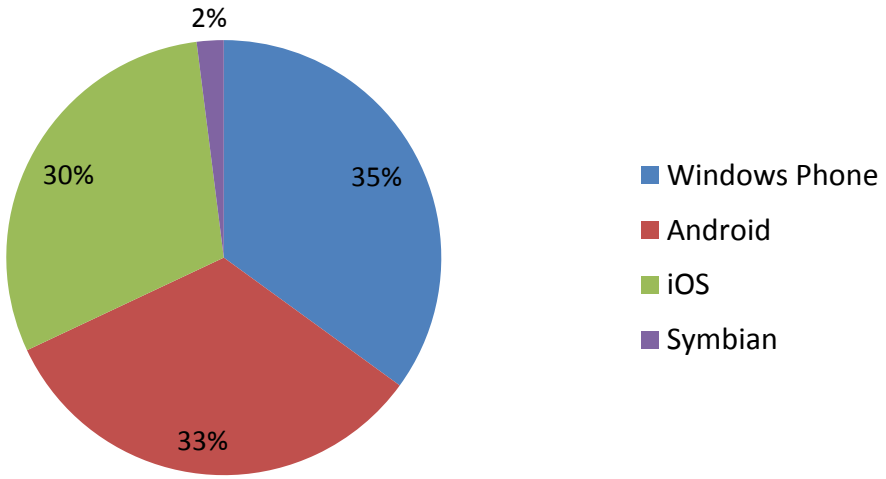
iOS (Internet Operating System) on Applen kehittämä käyttöjärjestelmä. Alun perin iOS kehitettiin iPhonea varten vuonna 2007, mutta myöhemmin se on laajentunut iPod Touchille, iPadille ja Apple TV:lle. Ensimmäinen iPhone oli samalla myös ensimmäinen kosketusnäytöllinen puhelin, joka ei vaatinut styluksen eli osoitinkynän käyttöä. (iOS: A visual history 2013.)

Viimeisin versio iOS:sta, iOS 7, julkaistiin 27. syyskuuta 2013. Uuden ulkoasun lisäksi käyttöjärjestelmä sai muun muassa Control Center -työkalupalkin, joka on Applen versio Android-laitteiden vastaavasta pika-asetusvalikosta. Toisin kuin Android ja Windows Phone, iOS-käyttöjärjestelmä on lisensoitu vain Applen tuotteille eikä sitä käytä muut kuin Applen valmistamat laitteet. (Mt.)

2.2.4 Windows Phone

Windows Phone on Microsoftin kehittämä mobiilikäyttöjärjestelmä, jonka ensimmäinen versio, Windows Phone 7, julkaistiin vuoden 2010 lopulla. Nokia on suurin Windows Phone -älypuhelinten valmistaja, muita ovat muun muassa HTC ja Samsung. Nokia käyttää älypuhelimiensa ensisijaisena käyttöjärjestelmänä Windows Phone 8:aa, joka on tällä hetkellä uusin Windows Phonen versio.

Taulukosta 1 voidaan todeta, että vaikka Windows Phone onkin maailmalla kolmanneksi käytetyin älypuhelinten käyttöjärjestelmä, sen markkinaosuus on varsin pieni (3,6 %). Suomessa taas Windows Phone on saavuttanut erityisaseman Nokian vuoksi. Market-Vision vuonna 2013 teettämän tutkimuksen mukaan se on Suomen suosituin mobiilikäyttöjärjestelmä. Windows Phonea tutkimuksen mukaan käytti 35 prosenttia suomalaisista, kun taas Androidin osuus oli 33 ja iOS:n 30 prosenttia (ks. kuvio 2).



Kuvio 2. Älypuhelinten käyttöjärjestelmien jakautuminen Suomessa (Windows Phone noussut -- 2013)

3 Mobiilisovellukset

3.1 Yleistä

Mobiilisovellukset tai -applikaatiot ovat sovelluksia, jotka on suunniteltu käytettäväksi nimenomaan älypuhelimilla ja tablet-tietokoneilla. Sovelluksia näihin laitteisiin ladataan yleensä laitteen käyttöjärjestelmän omalta jakelukanavalta. Tällaisia jakelukanavia ovat muun muassa Applen App Store, Googlen Google Play -kauppa ja Marketplace Windows Phone -älypuhelimille. Ladattavat sovellukset voivat olla käyttäjille joko ilmaisia tai maksullisia.

Kaikkia mobiilisovelluksia ei kuitenkaan ladata sovelluskaupoista, vaan osa sovelluksista toimii web-sivujen tapaan internetselaimen välityksellä. Nämä sovellukset ovat ns. web-sovelluksia. Sovelluskaupoista ladatut applikaatiot ovat useimmiten natiiveja sovelluksia, mutta osa niistä on toteutettu samaan tapaan kuin web-sovellukset, jolloin puhutaan hybridisovelluksista.

3.2 Natiivisovellukset

Natiivisovellukset ovat niitä sovelluksia, jotka ladataan laitteen omasta sovelluskaupasta ja asennetaan sen jälkeen laitteen muistiin. Natiivisovellukset ohjelmoidaan jokaiselle eri käyttöjärjestelmälle niiden vaatimilla ohjelmointikielillä. Esimerkiksi iPhone-sovellukset käyttävät Objective-C:tä, kun taas Android-laitteissa sovellukset kirjoitetaan useimmiten Javalla. Windows Phonen applikaatiot toteutetaan C#:lla tai Visual Basicilla. (Avola & Raasch 2013, 9.)

Natiivisovellukset toimivat laitteesta käsin eivätkä ole riippuvaisia internetistä vaan voivat toimia offline-tilassa. Natiivisovellukset pystyvät myös hyödyntämään laitteen sisäänrakennettuja ominaisuuksia, kuten kameraa, gps-paikannusta, kiihtyvyysmittaria, kompassia, kontaktilistaa jne. Lähes kaikki mobiilipelit ovat natiiveja sovelluksia. (Budiu 2013.)

3.3 Web-sovellukset

Web-sovellukset eivät ole oikeastaan perinteisiä sovelluksia vaan pikemminkin web-sivuja, jotka on optimoitu nimenomaan mobiililaitteille mukautuviksi. Web-sovellukset kirjoitetaan pääasiassa HTML5-, CSS3- ja JavaScript-ohjelmointikielillä verkkosivujen tapaan. Web-applikaatiot toimivat laitteen internetselaimesta käsin; käyttääkseen sovellusta käyttäjä kirjoittaa selaimen osoiteriville sovelluksen osoitteen aivan kuin minkä tahansa verkkosivun osoitteen. Näin ollen web-sovellukset vaativat useimmiten internetyhteyden käyttöä, mutta niitä voidaan myöhemmin käyttää myös offline-tilassa selaimen välimuistissa. Toisin kuin natiivisovellukset, web-applikaatiot eivät kykene käyttämään laitteen natiiveja ominaisuuksia eikä niitä voi asentaa laitteen muistiin samaan tapaan kuin natiivisovelluksia. Niitä voidaan kuitenkin tallentaa kirjainmerkkeinä selaimen muistiin. (Budiu 2013.)

3.4 Natiivi- vai web-sovellus?

Natiivi- ja web-sovelluksia on nykyisin hankala erottaa toisistaan pelkän ulkonäön perusteella (ks. kuvio 3). Web-sovellukset ovat kehittyneet viime vuosien aikana kilpailukykyisimmiksi ja ne saadaan näyttämään jo hyvin paljon natiiveilta sovelluksilta. Web-sovelluksien etuna on niiden helppo jakelu loppukäyttäjille; sama applikaatio toimii kaikilla laitteilla, joissa on asennettuna internetselain eikä sovellusta tarvitse tehdä erikseen eri päätelaitteille.

Kuvio 3. Facebookin natiivisovellus (vasemmalla) ja web-sovellus (oikealla)

Natiivisovellusten etuina web-applikaatioihin verrattuna ovat niiden suorituskyky sekä mahdollisuus hyödyntää laitteen natiiveja ominaisuuksia. Natiivisovelluksista on myös helpompi tehdä maksullisia sovelluskauppojen ansiosta. Vaikka natiivit sovellukset ovatkin tehokkaampia ja monipuolisempia kuin web-sovellukset on niiden kehittäminen hankalampaa ja kalliimpaa: jokainen eri käyttöjärjestelmä asettaa omat vaatimuksensa sovelluskehitykselle. Esimerkiksi iPhonelle ei voi tehdä natiivia sovellusta tietokoneella, jossa on Windows-käyttöjärjestelmä, vaan se vaatii Macin ja XCode-kehitysympäristön (ks. taulukko 2).

Taulukko 2. Natiivisovelluskehityksen vaatimukset eri käyttöjärjestelmissä (Frommel 2013)

Mobiilikäyttö-järjestelmä	**Tietokoneen käyttöjärjestelmä**	**Kehitysympäristö**	**Ohjelmointikieli**
Android	Windows, Mac, Linux	Eclipse	Java
iOS	Mac	XCode	Objective-C
Windows Phone	Windows	Visual Studio	C#, Visual Basic

Valittaessa toteutettavan sovelluksen tyyppiä kannattaa tehdä kartoitusta sovelluksen vaatimuksista. Tarvitseeko applikaation toimia sekä Android, iOS- että Windows Phone -laitteissa, täytyykö sen kyetä käyttämään jotakin laitteen natiivia ominaisuutta, esimerkiksi kameraa, ja millainen on sovelluskehitykseen varattu budjetti? Web-sovelluksen kehittäminen on yleensä helpompaa, nopeampaa ja halvempaa kuin natiivisovelluksen, mutta se ei ole suorituskyvyltään yhtä tehokas eikä se kykene hyödyntämään laitteen sisäänrakennettuja ominaisuuksia, kuten kameraa tai notifikaatioita eli ilmoituksia. Web-sovelluksen päivittäminen, ylläpito ja jakelu eri mobiilialustoille on vaivattomampaa, mutta web-sovelluksia ei saada esille eri käyttöjärjestelmien omiin sovelluskauppoihin kuten natiivisovelluksia. Natiivisovellus ja web-sovellus eivät kuitenkaan ole ainoat mobiilisovelluskehittäjän vaihtoehdot; näiden välimaastoon sijoittuvat hybridisovellukset, jotka yhdistelevät molempien sovellustyyppien ominaisuuksia. (Avola & Raasch 2013, 9-13.)

3.5 Hybridisovellukset

Hybridisovellukset ovat web-sovellusten ja natiivien applikaatioiden yhdistelmiä. Ne toteutetaan yleensä web-sovellusten tapaan web-tekniikoilla (HTML5, CSS3, JavaScript), mutta ne kykenevät hyödyntämään myös mobiililaitteen natiiveja ominaisuuksia. Tämän lisäksi hybridisovelluksia voidaan jakaa sovelluskauppojen, kuten App Storen tai Google Play -kaupan, kautta.

Vaikka hybridisovellukset toteutetaankin web-tekniikoilla, ne eroavat web-sovelluksista siinä, etteivät ne toimi mobiililaitteen internetselaimesta käsin, vaan natiivin sovelluksen sisällä on oma web view (web-ikkuna), joka hyödyntää laitteen selainmoottoria käsitelläkseen HTML-sisältöä. Parhaimmillaan hybridisovellus on yhdistelmä web- ja natiivisovellusten hyvistä puolista. Niiden maine on kuitenkin kärsinyt yritysten paketoitua tavallisia web-sovelluksia hybrideiksi saadakseen sovelluksensa sovelluskauppoihin. (Riippi 2013.)

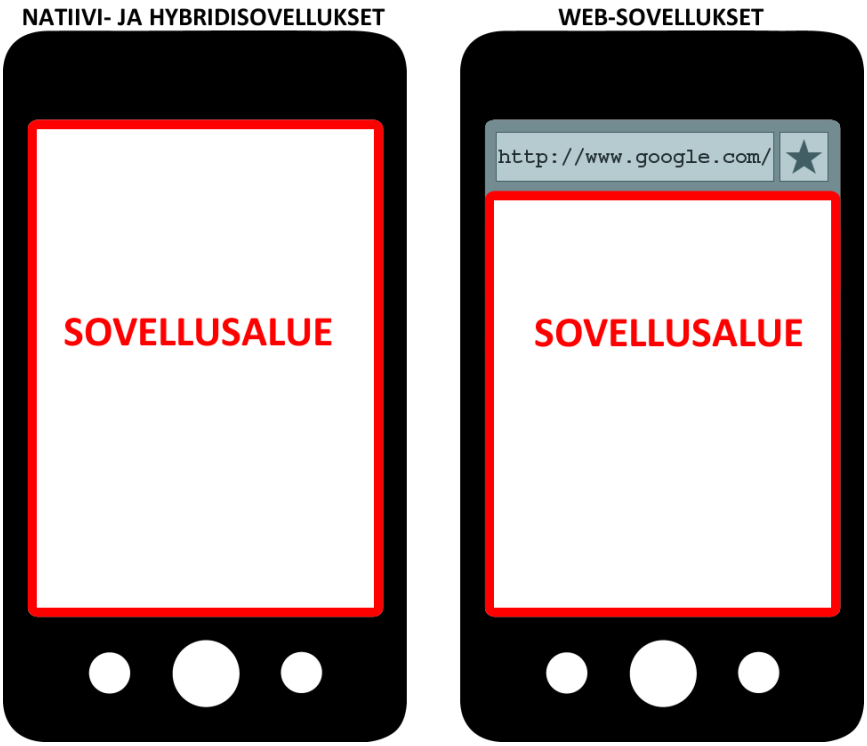
Hybridisovellusratkaisut ovat viime vuosina olleet kovassa nousussa. Yksi maailman johtavimmista tietotekniikka-alan tutkimus- ja konsultointiyrityksistä, Gartner, arvioi, että vuoteen 2016 mennessä jopa puolet mobiilisovelluksista olisi hybridejä. Yksi syy hybridisovellusten kasvavaan suosioon on niiden kehittämiseen suunniteltujen työkalujen yleistyminen; yksi tunnetuimmista on PhoneGap.

4 PhoneGap

4.1 Yleistä

PhoneGap on ohjelmistokehys hybridisovellusten toteuttamiseen HTML-, CSS- ja JavaScript-ohjelmointikielillä. Ohjelmistokehys eli framework on eräänlainen runko, jonka päälle ohjelmoija rakentaa sovelluksensa; se on tavallaan vaillinainen ohjelmisto, jota ohjelmoija täydentää tarkoituksenmukaisesti (Koskimies & Mikkonen 2005, 57-58). PhoneGap perustuu 100-prosenttisesti avoimeen lähdekoodiin ja kuuluu Apache Cordova -hankkeeseen (Trice 2012).

PhoneGap on yksi tunnetuimmista ja käytetyimmistä hybridisovellusten toteuttamiseen tarkoitetuista frameworkeista. Käyttöliittymäkerros PhoneGap-sovelluksissa muodostuu web view -ikkunasta, joka on 100-prosenttisesti laitteen näytön kokoinen. Tämä tarkoittaa sitä, ettei web view -ikkunassa näy selaimelle tyypillistä osoiteriviä vaan sovellus täyttää koko näytön natiivisovellusten tapaan (ks. kuvio 4). (Trice 2012.)



Kuvio 4. Sovellusalueet natiivi- ja hybridisovelluksissa vasemmalla ja web-sovelluksissa oikealla

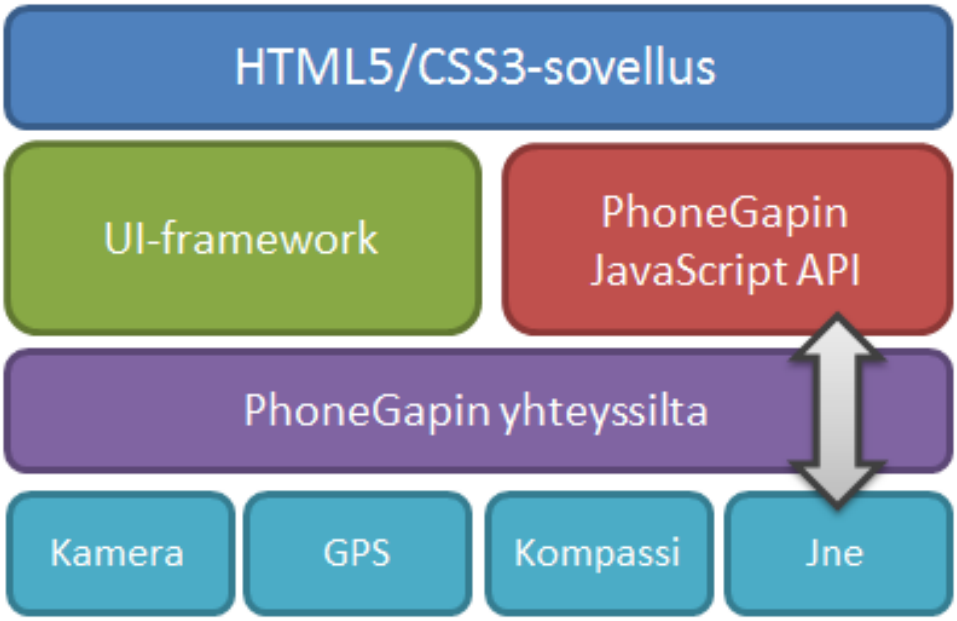
4.2 PhoneGapin historia

PhoneGapin ensimmäinen versio syntyi elokuussa 2008 San Franciscossa järjestetyssä iPhoneDevCamp-tapahtumassa, jossa pieni joukko kanadalaisen Nitobi-yrityksen työntekijöitä onnistui luomaan selainikkunan natiivin iPhone-sovelluksen sisään. Alkuun Nitobin työryhmä keskittyi nimenomaan iPhoneen, sillä osaavia Objective-C-kehittäjiä oli web-kehittäjiin verrattuna hyvin vähän. Ensimmäinen vakaa versio PhoneGapista julkaistiin helmikuussa 2009, ja se sisälsi tuen iPhonen lisäksi Androidille ja BlackBerrylle. (Taft 2009.)

Huhtikuussa 2009 PhoneGap voitti People's Choice -palkinnon Web 2.0 Expo -tapahtumassa, jonka myötä PhoneGap tuli suuren yleisön tietoisuuteen (Davis 2009). Loppuvuodesta 2009 PhoneGap alkoi tukea myös WindowsMobilea ja Nokia S60-puhelimia. Lokakuussa 2011 Adobe Systems osti Nitobin, jolloin PhoneGapin lähdekoodi luovutettiin Apache Software Foundationille. PhoneGapin Apache-projektin nimeksi tuli alkuun Apache Callback, joka kuitenkin myöhemmin muutettiin Apache Cordovaksi. PhoneGap on siis Apache Cordovan yksi jakeluosuus; Apache Cordova toimii ikään kuin moottorina PhoneGapille samaan tapaan kuin WebKit Google Chromelle ja Safarille (Leroux 2012). PhoneGap on tällä hetkellä menossa versiossa 3.4.0, ja se on yhteensopiva muun muassa iPhonen, Androidin sekä Windows Phone 7:n ja 8:n kanssa.

4.3 PhoneGapin sovellusarkkitehtuuri

Sovellusarkkitehtuuri sisältää sovelluksen tai järjestelmän osat, niiden suhteet toisiinsa ja ympäristöön sekä sovelluksen suunnittelua ja jatkokehitystä ohjaavat periaatteet (Koskimies & Mikkonen 2005, 18). PhoneGap-sovellukset rakentuvat kuvion 5 (s. 16) mukaisesti: PhoneGapin yhteyssilta toimii JavaScript-rajapinnan ja laitteen natiivien ominaisuuksien välissä. Sovelluksien toteuttamiseen voidaan myös käyttää erillistä käyttöliittymäkehystä (esim. jQuery Mobile tai Sencha Touch), joka helpottaa käyttöliittymän rakentamista. (Ghatol & Patel 2012, 18.)



Kuvio 5. PhoneGapin sovellusarkkitehtuuri (Ghatol & Patel 2012, 18)

PhoneGap-sovelluksen hakemistorakenne koostuu viidestä kansiosta (hooks, merges, platforms, plugins ja www) ja config-xml-asetustiedostosta, jonne määritetään sovelluksen tietoja, kuten nimi, tekijä, versionumero, kohdealusta jne. Hooks-kansio sisältää skriptejä eli komentosarjoja, joilla voidaan muokata PhoneGapin komentoja. Merges-hakemistoon voidaan määrittää tietyillä laitteilla käytettäviä ominaisuuksia. Esimerkiksi iOS-laitteelle voitaisiin merges-hakemistoon määrittää back- eli takaisin-painike kun taas Android-laitteessa takaisin-toiminto voidaan hoitaa laitteen omalla natiivilla painikkeella. (Moore 2013.)

Platforms-hakemisto sisältää kunkin kohdealustan mukaiset koontitiedostot. Plugins-hakemistoon voidaan tallentaa erilaiset sovelluksessa käytettävät liitännäiset eli pluginit. Itse sovelluslogiikka on kuitenkin www-kansiossa: se sisältää hakemistot JavaScriptille, kuville ja CSS:lle sekä index.html-tiedoston, johon sovelluksen html-koodi kirjoitetaan. (PhoneGap Documentation 2014.)

4.4 PhoneGapin ohjelmointirajapinnat

4.4.1 Yleistä

PhoneGap tarjoaa ohjelmistokehittäjille ohjelmointirajapinnan (Application programming interface, API), jonka avulla päästään käsiksi mobiililaitteen natiiveihin omi-

naisuuksiin JavaScriptillä. Rajapintojen avulla eri ohjelmat voivat niin sanotusti keskustella keskenään eli tehdä pyyntöjä ja vaihtaa tietoja. PhoneGapin tapauksessa sovelluskehittäjä rakentaa sovelluksensa toimintalogiikan JavaScriptillä ja PhoneGap toimii tavallaan siltana sovelluksen ja laitteen välillä. (Trice 2012.)

PhoneGapin nykyinen versio (3.4.0) tukee hyvin uusimpia mobiililaitteita. Taulukosta 3 voidaan todeta, että sekä iOS, Android että Windows Phone 7 ja 8 ovat hyvin tuettuja. Vain kompassille ei löydy tukea vanhemmilla iPhone-älypuhelimilla. Monia taulukosta puuttuvia ominaisuuksia, kuten kalenteria, viivakoodinlukijaa, bluetooth-yhteyttä, push-ilmoituksia tai Facebook-tilin yhdistämistä sovellukseen, voidaan hyödyntää erillisten liitännäisten avulla (Traeg 2014).

Taulukko 3. PhoneGapin tuki eri ohjelmointirajapinnoille tämän hetken suosituimmissa mobiilikäyttöjärjestelmissä (PhoneGap 2014)

	iPhone / iPhone 3 G	iPhone 3 GS tai uudempi	Android	Windows Phone 7 ja 8
Ilmoitukset	✓	✓	✓	✓
Kamera	✓	✓	✓	✓
Kiihtyvyysmittari	✓	✓	✓	✓
Kompassi	X	✓	✓	✓
Media	✓	✓	✓	✓
Muisti	✓	✓	✓	✓
Sijainti	✓	✓	✓	✓
Tiedosto	✓	✓	✓	✓
Yhteys	✓	✓	✓	✓
Yhteystiedot	✓	✓	✓	✓

PhoneGap-versiossa 3.4.0 sovelluksessa tarvittavat rajapinnat lisätään sovellukseen liitännäisinä eli plugineina. Ohjelmointirajapintoja kutsutaan muutamaa poikkeusta lukuun ottamatta aina seuraavasti:

```
navigator.rajapinnannimi.rajapinnanmetodi(kutsuOnnistui, kut-
suEpäonnistui, parametrit)
```

”Rajapinnannimi”-kohtaan tulee kutsuttavan rajapinnan nimi, esimerkiksi camera tai notification. Rajapinnan metodi voisi kamerarajapinnan tapauksessa olla esimerkiksi getPicture, joka avaa mobiililaitteen oletuskameran kuvan ottamista varten. Suurin osa PhoneGapin rajapintakutsuista sisältävät takaisinkutsu- eli callback-funktiot sekä onnistuneelle että epäonnistuneelle rajapintakutsulle. Mikäli kutsu onnistuu, voidaan toteuttaa haluttu toiminto, jos taas kutsu epäonnistuu, voidaan käyttäjälle näyttää virheilmoitus. Takaisinkutsufunktioiden lisäksi rajapintakutsuille voidaan antaa useita erilaisia parametreja. Kamerarajapinnan getPicture-kutsulle voitaisiin parametreina viedä esimerkiksi kuvan laatu ja koko. (PhoneGap Documentation 2014.)

4.4.2 Ilmoitukset

Ilmoituksilla eli notifikaatioilla tarkoitetaan mobiililaitteen erilaisia viestejä, joita sovellus lähettää käyttäjälle. Ilmoituksiin luetaan dialogi- eli ponnahdusikkunat, äänimerkit ja laitteen värinätoiminnot. PhoneGapin ilmoitusrajapinta käyttää ilmoitusten äänimerkkinä aina käyttäjän itse laitteelle asettamansa vakioilmoitusmerkkiääntä. Tavallisen ponnahdusikkunatyyppisen ilmoituksen saa luotua esimerkiksi seuraavalla tavalla:

```
function showAlert() {
        navigator.notification.alert(
            'Voitit pelin!',        // viesti
             alertDismissed,        // callback-funktio
            'Onneksi olkoon',       // otsikko
            'Hyväksy'               // painikkeen teksti
        );
}
```

4.4.3 Kamera

Kamerarajapinnan avulla voidaan ottaa kuva laitteen oletuskameralla tai vaihtoehtoisesti hakea kuva laitteen kuva-albumeista. Kuvasta välitetään sovellukselle joko kuvan URI- eli Uniform Resource Identifier -osoite tai base64-koodattu String- eli merkkijonomuuttuja. Joko kameralla juuri kuvattua tai kuva-albumeista haettua kuvaa voidaan sovelluksessa käsitellä monin eri tavoin: kuva voidaan esimerkiksi näyttää sovelluksessa, tallentaa laitteen muistiin tai lähettää etäpalvelimelle. Seuraavalla koodilla käyttäjälle avataan oletuskameranäkymä, jossa käyttäjä voi ottaa kuvan. Otetusta kuvasta palautetaan sovellukseen sen URI-osoite.

```
navigator.camera.getPicture(onSuccess, onFail, {
  quality: 75,
  destinationType: Camera.DestinationType.FILE_URI
});

// Kuva haettiin kameralta onnistuneesti
function onSuccess(imageURI) {
    var image = document.getElementById('kameranKuva');
    image.src = imageURI;
}

// Kuvan haku kameralta epäonnistui
function onFail(message) {
    alert('Tapahtui virhe: ' + message);
}
```

Kameran getPicture-funktiolle voidaan funktiokutsussa syöttää useita erilaisia parametreja. Näitä ovat muun muassa kuvan laatu, joka ilmoitetaan numeroarvolla 0-100 väliltä, jossa arvo 100 tarkoittaa parasta mahdollista laatua. Tärkeimpiä parametreja ovat kuitenkin palautettavan kuvan tyyppi (merkkijono vai kuvan osoite) ja kuvan lähde eli otetaanko kuva kameralla vai haetaanko se laitteen muistista. (PhoneGap Documentation 2014.)

4.4.4 Kiihtyvyysmittari

Kiihtyvyysmittari toimii liiketunnistimen tapaan: se reagoi laitteen kallistumiseen, sijainnin muutoksiin ja kiihtyvyyteen vertaamalla laitteen nykyistä sijaintia ja suuntaa kolmessa ulottuvuudessa x-, y- ja z-akselien suhteen. Kiihtyvyysmittarirajapinnalla on kolme metodia: getCurrentAcceleration, joka palauttaa laitteen nykyisen sijainnin koordinaatit x-, y- ja z-akselien suhteen, watchAcceleration, joka tarkkailee laitteen kiihtyvyyden muutoksia ja palauttaa kiihtyvyyvyyden arvoja tasaisin väliajoin sekä clearWatch, jota käytetään keskeyttämään kiihtyvyyden tarkkailu käyttämällä kiihtyvyydelle annettua tunnusta eli id:tä. Seuraavassa esimerkissä käyttäjälle ilmoitetaan laitteen senhetkinen kiihtyvyys x-, y- ja z-akseleiden suhteen:

```
function onSuccess(acceleration) {
    alert('Kiihtyvyys X: ' + acceleration.x + '\n' +
          'Kiihtyvyys Y: ' + acceleration.y + '\n' +
          'Kiihtyvyys Z: ' + acceleration.z + '\n' +
          'Aikaleima: '    + acceleration.timestamp + '\n');
};
```

```
function onError() {
    alert('Tapahtui virhe');
};

navigator.accelerometer.getCurrentAcceleration(onSuccess,
onError);
```

4.4.5 Kompassi

Kompassirajapinta mahdollistaa laitteen suunnan tarkkailun. Kompassi tunnistaa, mihin suuntaan laite osoittaa. Suunta palautetaan astelukuna väliltä 0-359.99, jossa 0 tarkoittaa pohjoista. Laitteen suuntaa voidaan tarkkailla tasaisin väliajoin samaan tapaan kuin kiihtyvyyttä. Tämän lisäksi voidaan asettaa raja, jonka ylityttyä käyttäjälle näytetään ilmoitus sijainnin muuttumisesta. Laitteen suunnan seuranta kolmen sekunnin välein saadaan aikaan seuraavasti:

```
function onSuccess(heading) {
    var element = document.getElementById('suunta');
    element.innerHTML = 'Suunta: ' + heading.magneticHeading;
};

function onError(compassError) {
    alert('Tapahtui virhe: ' + compassError.code);
};

var options = {
    frequency: 3000
}; // Tarkkaillaan 3 sekunnin välein

var watchID = navigator.compass.watchHeading(onSuccess, onError,
options);
```

4.4.6 Media

Mediarajapinta tarjoaa mahdollisuuden tallentaa ja toistaa äänitiedostoja sovelluksessa. Mediarajapinnalla on paljon erilaisia metodeja: mediasoittimille yleiset play-, stop- ja pause, äänenvoimakkuuden säätö, uuden äänitiedoston nauhoitus sekä nykyisestä äänitiedostosta voidaan palauttaa sen kesto tai kohta, jossa ollaan menossa toistettaessa ääntä. Esimerkki äänitiedoston toistamisesta sovelluksessa:

```
// Aloita äänitiedoston toistaminen
function playAudio(url) {
    // Toista äänitiedosto annetusta osoitteesta
    var aanitiedosto = new Media(url,
        // callback-funktio
        function () {
```

```
            console.log("Äänitiedoston toisto onnistui");
        },
        // callback-funktio virhetilanteissa
        function (err) {
            console.log("Tapahtui virhe: " + err);
        }
    );
    // Toista äänitiedosto
    aanitiedosto.play();
}
```

4.4.7 Muisti

Muistirajapinnan avulla sovelluksessa voidaan tallentaa erilaisia tietoja muistiin; sovelluksessa voi olla vaikkapa mahdollisuus säätää joitakin asetuksia ja nämä asetukset voidaan sitten tallentaa, jotta sovellus muistaa valitut asetukset seuraavalla kerralla. Menetelmiä tiedon tallentamiseen on neljä: LocalStorage, WebSQL, IndexedDB ja erilaiset liitäntä- eli plugin-ratkaisut. (PhoneGap Documentation 2014.)

LocalStorage eli paikallismuisti on HTML5-teknologia, jonka avulla voidaan tallentaa tietoa paikallisesti selaimeen avain/arvo-pareina. LocalStorageen tallennettu tieto säilyy aina niin pitkään kunnes selain poistetaan tai asennetaan uudelleen. WebSQL tarjoaa monipuolisemman tietokantaratkaisun kuin LocalStorage; WebSQL-tietokantaan voidaan kohdistaa sql-kyselyjä, joilla tiedonhaku muistista tarkentuu. Tietokantaan voidaan myös tallentaa monipuolisempaa tietoa kuin selaimen paikallismuistiin. Windows Phone -älypuhelimet eivät kuitenkaan tue WebSQL-tietokantoja, mutta niissä WEBSQL voidaan korvata IndexedDB:llä. IndexedDB tarjoaa enemmän ominaisuuksia kuin paikallismuisti mutta vähemmän kuin WebSQL. Näiden kolmen vaihtoehdon lisäksi muistirajapinnan avulla voidaan tallentaa tietoja laitteen paikalliseen välimuistiin erilaisten liitännäisten eli pluginien avulla. Esimerkki paikallismuistiin tallentamisesta:

```
// Haetaan lomakekentistä muuttujien arvot
var nameVal = $('#name').val();
var zipcodeVal = $('#zipcode').val();
var cityVal = $('#city').val();

// Tallennetaan tiedot paikallismuistiin
window.localStorage.setItem('name' , nameVal);
window.localStorage.setItem('zipcode' , zipcodeVal);
window.ocalStorage.setItem('city' , cityVal);
```

4.4.8 Sijainti

Sijaintirajapinta tutkii laitteen senhetkistä sijaintia gps-paikannuksen (Global Positioning System) avulla ja palauttaa sijainnin leveys- (latitude) ja pituusasteen (longitude). Sijainti voidaan myös päätellä erilaisten verkon signaalien, kuten IP-osoitteen, RFID:n, WiFi- ja Bluetooth-yhteyden MAC-osoitteen tai GSM/CDMA-solutunnuksen, avulla. Tällöin ei tosin voida taata, että funktion palauttama arvo on laitteen todellinen sijainti. Kuten kiihtyvyyttä ja suuntaa, myös sijaintia voidaan sekä tarkkailla tasaisin väliajoin tai käyttäjälle voidaan palauttaa laitteen nykyinen sijainti leveys- ja pituusasteen avulla seuraavasti:

```
// Laitteen sijainti paikallistettiin onnistuneesti
var onSuccess = function(position) {
    alert('Leveysaste: ' + position.coords.latitude + '\n' +
          'Pituusaste: ' + position.coords.longitude + '\n');
};

// Laitteen sijainnin paikallistaminen epäonnistui
function onError(error) {
    alert('Virhekoodi: '     + error.code    + '\n' +
          'Tapahtui virhe: ' + error.message + '\n');
}

navigator.geolocation.getCurrentPosition(onSuccess, onError);
```

4.4.9 Tiedosto

Tiedostorajapinta seuraa www-standardien kehittäjän ja ylläpitäjän W3C:n (World Wide Web Consortium) määrityksiä. Tiedostorajapinnan avulla sovelluksella voidaan sekä selata että lukea mobiililaitteen hakemistoja ja tiedostoja. Tiedostoilla on useita eri ominaisuuksia, joita voidaan tarkastella tiedostorajapinnan avulla. Näihin kuuluvat muun muassa tiedostonimi, tiedoston polku eli sijainti laitteessa, tiedostotyyppi, päivämäärä, jolloin tiedostoa on viimeksi muokattu ja tiedostokoko. Seuraavassa esimerkissä tiedostorajapintaa käytetään hakemiston uudelleennimeämiseen ja siirtämiseen uuteen paikkaan hakemistopuussa:

```
// Hakemistonsiirto onnistui
function success(entry) {
    console.log("Uusi polku: " + entry.fullPath);
}

// Hakemistonsiirto epäonnistui
function fail(error) {
```

```
    alert("Tapahtui virhe: " + error.code);
}

// Hakemistonsiirtofunktio
function moveDir(entry) {
    var parent = document.getElementById('parent').value,
        parentName = par-
ent.substring(parent.lastIndexOf('/')+1),
        newName = document.getElementById('newName').value,
        parentEntry = new DirectoryEntry(parentName, parent);

    // Siirtää ja uudelleennimeää hakemiston
    entry.moveTo(parentEntry, newName, success, fail);
}
```

4.4.10 Yhteys

Yhteysrajapinta tarjoaa tietoa mobiililaitteen puhelinverkko- ja wifi-yhteyksistä ja kykenee tarkistamaan onko laite kytkettynä internetiin. Yhteysrajapinnan avulla voidaan selvittää käytetyn yhteyden tyyppi ja tarkkailla internetyhteyden tilaa; sovellus voi esimerkiksi ilmoittaa käyttäjälle, jos internet-yhteys katkeaa seuraavasti:

```
// Tarkkaillaan yhteyttä
document.addEventListener("offline", onOffline, false);

// Ilmoitetaan käyttäjälle, että laite on offline-tilassa
function onOffline() {
    alert("Laite on offline-tilassa.”);
}
```

4.4.11 Yhteystiedot

Yhteystietorajapinnalla sovellus voi käyttää laitteen kontaktilistaa; listaan voidaan luoda uusia kontakteja, vanhoja voidaan poistaa tai kopioida ja kontakteja voidaan näyttää sovelluksessa. Yhteystietolistaan voidaan lisätä uusi yhteystieto seuraavasti:

```
// Kontaktin tallentaminen onnistui
function onSuccess(contact) {
    alert("Kontakti tallennettu.");
};

// Kontaktin tallentaminen epäonnistui
function onError(contactError) {
    alert("Tapahtui virhe: " + contactError.code);
};
```

```
// Luodaan uusi kontaktiolio
var contact = navigator.contacts.create();
contact.displayName = "Esimies";
contact.nickname = "Pomo";
var name = new ContactName();
name.givenName = "Matti";
name.familyName = "Meikäläinen";
contact.name = name;

// Tallennetaan luotu kontakti kontaktilistaan
contact.save(onSuccess,onError);
```

4.5 PhoneGapin käyttöönotto

PhoneGapin dokumentaatiosta käy ilmi, että ennen PhoneGapin asentamista tulee asentaa kaikkien haluttujen kohdealustojen omat SDK:t. SDK (Software Development Kit) on ohjelmistopaketti, joka mahdollistaa sovelluskehityksen tietyllä alustalla. Tyypillisesti SDK sisältää yhden tai useamman ohjelmointirajapinnan, erilaisia ohjelmointityökaluja sekä dokumentaation. Kehitettäessä PhoneGap-sovelluksia lokaalisti, tulee kehitysympäristö valita kunkin kohdealustan mukaan. PhoneGapin oma Command-Line Interface (CLI) eli komentorivikäyttöliittymä tukee seuraavia kombinaatioita:

- iOS (Mac)
- Android (Mac, Linux, Windows)
- Windows Phone (Windows)

Kehitysympäristön lisäksi PhoneGapin asennus vaatii Node.js:n. Node.js on kevyt JavaScript-sovellusalusta web-palvelujen kehitykseen palvelinpäässä. Node.js:n voi ladata ja asentaa ilmaiseksi Node.js:n virallisilta verkkosivuilta tai suoraan tietokoneen komentoriviltä. Kun Node.js on asennettu onnistuneesti, voidaan asentaa itse PhoneGap komentoriviltä seuraavasti:

```
npm install -g phonegap
```

Asennuskomento voi tarvita toimiakseen etuliitteen “sudo”. Tällöin asennus suoritetaan järjestelmänvalvojana. Kun PhoneGap on onnistuneesti asennettu, voidaan ensimmäinen sovellus luoda komentoriviltä käskyllä

```
phonegap create hello com.example.hello HelloWorld
```

Sovelluksen luomiskäskyssä “hello” on projektille luotavan kansion nimi. Loput kaksi parametria ovat vapaaehtoisia mutta suositeltavia: ensimmäinen on projektin käänteisen domainin -tyylinen tunniste ja toinen sovelluksen nimi. Kumpaakin vapaaehtoista parametria voidaan muokata myös jälkeenpäin config.xml-tiedostosta. Kaikki sovellusta käsittelevät komennot täytyy tämän jälkeen ajaa komentoriviltä sovelluksen hakemistosta käsin. Sovellus ”buildataan” eli kootaan ajoa varten valmiiksi seuraavasti:

```
phonegap build android
```

Tässä käskyssä sanan “android”-tilalle laitetaan haluttu kohdealusta, jolle sovellus käännetään. Buildauksen jälkeen sovellusta voidaan testata emulaattorissa komennoilla

```
phonegap install android
phonegap run android
```

Ensimmäisellä komennolla asennetaan halutun kohdealustan emulaattori ja toisella ajetaan sovellus tässä emulaattorissa. (PhoneGap Documentation 2014.)

4.6 PhoneGap Build

4.6.1 Yleistä

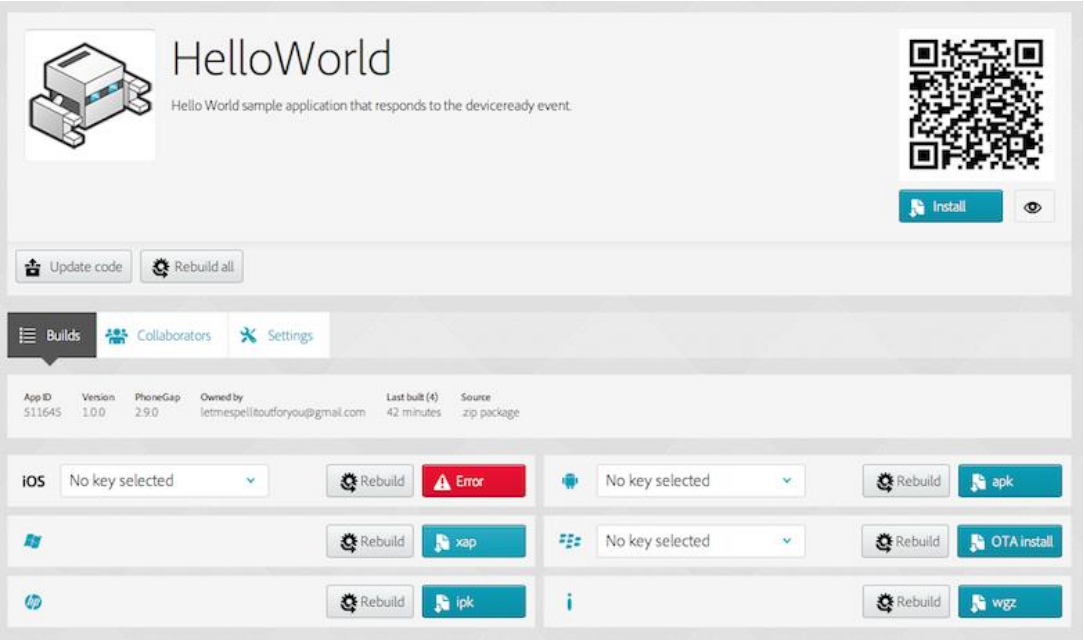
PhoneGap tarjoaa sovelluskehittäjille mahdollisuuden luoda laitteen natiiveja ominaisuuksia hyödyntäviä mobiilisovelluksia web-tekniikoilla ja paketoida nämä sovellukset sovelluskauppoihin jaettavaksi. Kuitenkin jokainen eri päätelaite asettaa omat vaatimuksensa sovelluksen kehitysympäristölle ja näin ollen saman sovelluksen monistaminen eri päätelaitteille hankaloituu vaikka suurimman osan koodista voisikin käyttää uudelleen. Tätä prosessia helpottamaan kehitetty PhoneGap Build tarjoaa vaihtoehtoisen laiteriippumattoman tavan sovellusten paketoimiseen.

PhoneGap Build on Adoben kehittämä pilvipalvelu PhoneGap-sovellusten paketointiin. Sovelluksen tiedostot voidaan joko ladata PhoneGap Buildiin zip-pakettina tai ne voidaan hakea sovelluskehittäjän Git- tai SVN-talletuspaikasta (repository). PhoneGap Build pakkaa tiedostot valmiiksi sovellukseksi. PhoneGap Build tukee Android-, iOS- ja Windows Phone 8 -laitteita. (PhoneGap Build Documentation 2014.)

4.6.2 Sovellusten pakkaaminen PhoneGap Buildin avulla

PhoneGap Buildissa on uudelle käyttäjälle kolme eri vaihtoehtoa: täysin ilmainen versio, joka sisältää yhden yksityisen sovelluksen ja loputtoman määrän avoimen lähdekoodin sovelluksia, joiden tiedostot tulee hakea julkiselta GitHub-talletuspaikasta. Maksullisessa versiossa ja Adoben Creative Cloud -jäsenillä yksityisten sovellusten määrä kasvaa 25:een. PhoneGap Buildiin kirjaudutaan sisään joko GitHubin tunnuksilla tai Adobe ID:llä. (PhoneGap Build Documentation 2014.)

PhoneGap Buildiin ladataan PhoneGap-sovellusprojektista vain www-kansio ja config.xml-tiedosto. PhoneGap Buildiin kirjautumisen jälkeen valitaan halutaanko pakata yksityinen vai julkinen sovellus. Yksityistä sovellusta varten sovelluksen projektitiedostot pakataan zip-muotoon, joka ladataan sitten PhoneGap Buildiin pakattavaksi. Julkisen avoimen lähdekoodin sovelluksen tiedostot haetaan suoraan GitHubista. Kun projektitiedostot on joko ladattu zip-pakettina tai haettu GitHub-versiohallinnasta PhoneGap Buildiin, ne pakataan kaikille niille kohdealustoille, jotka sovelluskehittäjä on määrittänyt sovelluksensa config.xml-tiedostoon. PhoneGap Buildin yksittäisen sovelluksen sivulla voidaan tarkastella sovelluksen tietoja, joita ovat muun muassa sovelluksen nimi, kuvake, versionumero, käytetyn PhoneGap-ohjelmistokehyksen versionumero, edellisestä sovelluspakkauksesta kulunut aika (ks. kuvio 6, s. 27).



Kuvio 6. PhoneGap Buildin käyttöliittymä

Valmista pakattua sovellusta voi testata mobiililaitteella lukemalla sovelluksen QR-koodin tai avaamalla laitteen selaimessa sovelluksen asennuslinkin. iOS-sovellukset vaativat kuitenkin iOS-sovelluskehitysohjelman lisenssitiedot eikä iOS-sovelluksen testaaminen mobiililaitteella onnistu ilman niitä. Android-sovelluksia voidaan kokeilla niitä tukevilla mobiililaitteilla ilman sovellusavainta: sitä tarvitaan vasta julkaisuvaiheessa. Androidille yksityisen sovellusavaimen tekeminen on ilmaista. iOS-kehittäjällä on oltava Applen kehittäjälisenssi, joka maksaa 99 dollaria vuodessa (Apple Developer Programs 2014). Tämän lisäksi tarvitaan Mac-käyttöjärjestelmän tietokone ja Xcode-ohjelma, jolla sovelluskehitysohjelman lisenssitiedot voidaan purkaa. Kun lisenssitiedot on purettu, voidaan sovelluskehitystä jatkaa millä tahansa käyttöjärjestelmällä. Lisenssitietojen lisäksi iOS-sovellus vaatii iOS-avaimen AppStore-jakelua varten. (Ghatol & Patel 2012, 116-123.)

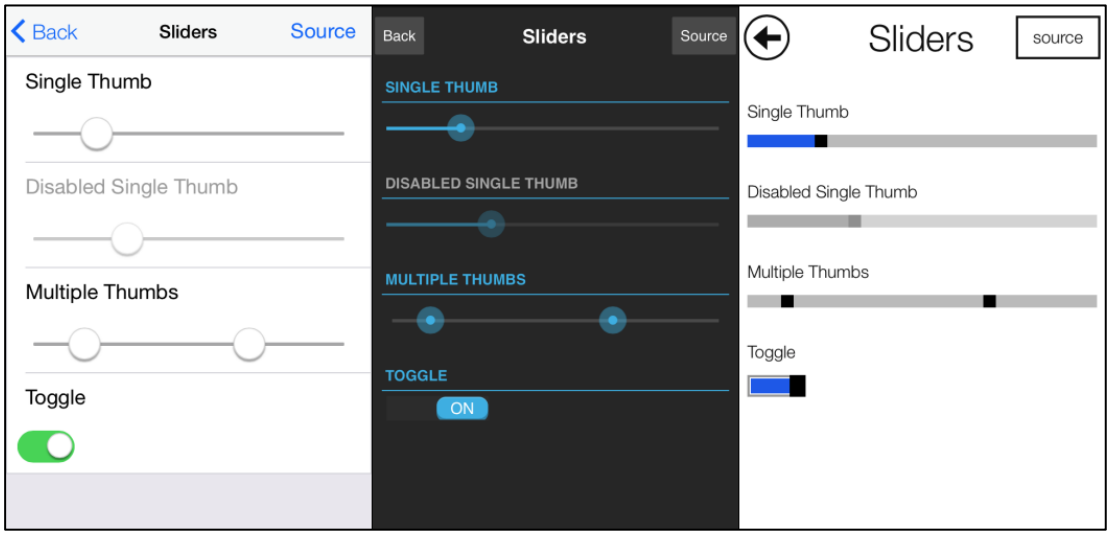
Windows Phone 8 -sovelluksia varten täytyy rekisteröityä Windows Phone -sovelluskehittäjäksi ja avata Windows Phone 8 -mobiililaite sovelluskehittäjän tunnuksilla sovelluskaupan ulkopuolisia applikaatioita varten. Tämän jälkeen PhoneGap Build:lla pakattua sovellusta voidaan testata Windows Phone 8 -laitteella. Sovelluskauppajakelua varten sovellus tarvitsee vielä julkaisijatunnuksen. Tätä varten täytyy olla sovelluskehittäjän tunnukset: pienten yritysten ja yksityishenkilöiden tunnus maksaa 14 euroa ja suurempien yritysten tunnus 75 euroa vuodessa. Kun sovelluksen

julkaisijatunnus on ladattu PhoneGap Buildiin, voidaan Windows Phone 8 -sovellus julkaista Windows Phone Marketplace -sovelluskaupassa. (PhoneGap Build Documentation 2014.)

5 Sencha Touch

5.1 Yleistä

Sencha Touch on mobiililaitteille suunnattujen web-sovellusten toteuttamiseen kehitetty JavaScript-koodikirjasto ja -framework. Sencha Touchilla toteutetut käyttöliittymät näyttävät ja tuntuvat hyvin pitkälti kullekin päätelaitteelle yksilöllisesti suunnitelluilta natiivisovelluksilta; Sencha Touchilla on omat teemansa muun muassa Androidille, iOS:lle ja Windows Phone 8:lle (ks. kuvio 7). (Sencha Touch Build Mobile Web Apps with HTML5 2014.)



Kuvio 7. Sencha Touchin teemat iOS-, Android- ja Windows Phone 8 -käyttöjärjestelmissä (Sencha Touch Build Mobile Web Apps with HTML5 2014)

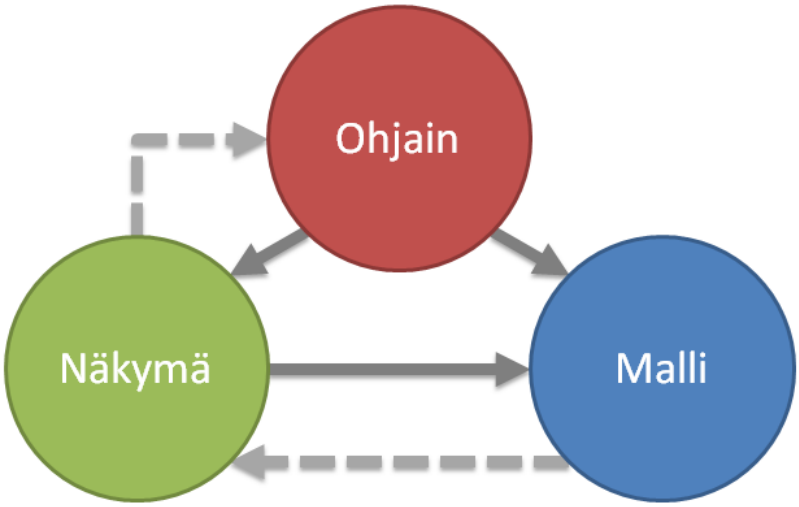
Ensimmäinen versio Senha Touchista julkaistiin heinäkuussa 2010, kun ExtJS-, jQTouch- ja Raphaël-JavaScript-kirjastot yhdistettiin yhdeksi paketiksi, ja se sisälsi tuen Androidille ja iOS:lle. Uusin versio, Sencha Touch 2.3.1, julkaistiin marraskuussa 2013 ja se on suunniteltu toimimaan Androidin ja iOSin lisäksi Windows Phone 8 ja Windows 8 -käyttöjärjestelmissä. (Kosmaczewski 2013, 2-4.)

5.2 Sencha Touch -sovelluksen rakenne

5.2.1 Sovellusarkkitehtuuri

Sencha Touch perustuu MVC-arkkitehtuuriin. Lyhenne MVC tulee sanoista Model eli malli, View eli näkymä ja Control eli ohjain. MVC-sovellusarkkitehtuurin perusajatuksena on erottaa käyttöliittymäkerros varsinaisesta sovelluslogiikasta ja sovelluksessa käytettävästä datasta. Kun käyttöliittymä on erillään muusta sovelluksesta, sitä voidaan helposti muuttaa tai se voidaan vaihtaa jopa täysin uuteen. (Koskimies & Mikkonen 2005, 142.)

MVC-sovellusarkkitehtuuria käyttävissä sovelluksissa sovellus jaetaan kolmeen osaan: malleihin, näkymiin ja ohjaimiin (ks. kuvio 8). Malli käsittää sovelluksessa käytettävän dataan liittyvät toiminnot: mitä dataa sovelluksessa hyödynnetään ja miten sitä käsitellään. Näkymät määrittävät käyttöliittymän eli sovelluksen ulkoasun ja sen, miten mallien tieto esitetään sovelluksessa. Ohjain toimii tavallaan siltana mallien ja näkymien välissä; se ottaa käskyjä käyttäjältä ja muuttaa näkymää tai mallia niiden mukaan. (Mt.)



Kuvio 8. MVC-sovellusarkkitehtuuri

Sencha Touch -sovellusten arkkitehtuuri eroaa tavallisesta MVC-mallista siinä, että mallien, ohjainten ja näkymien lisäksi se voi sisältää myös profiileja (profile) ja varastoja (store). Profiileilla voidaan määrittää miltä sovellus näyttää eri päätelaitteilla, kuten tableteilla ja älypuhelimilla. Varastot vastaavat puolestaan datan lataamisesta sovellukseen ja datan järjestämisestä mallin avulla. (Vino 2014.)

5.2.2 Luokkajärjestelmä

Sencha Touchia kirjoitetaan hyvin samaan tapaan kuin olio-ohjelmointikieliä; sen syntaksi perustuu olio-ohjelmoinnin tapaan luokille ja niiden instansseille eli esiintymille. Sencha Touchin luokkajärjestelmä perustuu kahdelle pääfunktiolle: Ext.define() ja Ext.create(), joista ensimmäisen avulla voidaan määrittää uusia luokkia ja kaikki niiden määrittämiseen tarvittavat tiedot, ja toisella luodaan olemassa olevista luokista uusia instansseja (Kosmaczewski 2013, 23-27). Yksinkertainen kontaktilistaluokka voitaisiin luoda esimerkiksi seuraavasti:

```
Ext.define('HelloWorld.view.ContactList', {
      extend: 'Ext.dataview.List',
      xtype: 'contactlist',
      config: {
                  /* Luokan määritykset */
      }
});
```

Yllä olevassa koodissa määriteltiin uusi luokka HelloWorld.view.ContactList, joka luodaan laajentamalla Sencha Touchin olemassa olevaa Ext.dataview.List-luokkaa. Tästä kontaktilistaluokasta voidaan luoda uusi instanssi seuraavasti:

```
var list = Ext.create('HelloWorld.view.ContactList', {
      items: [
                  /* Listaelementit */
      ],
      title: 'Yhteystiedot',
      /* Ym. parametrit eroteltuina pilkulla */
});
```

Yksi Sencha Touchin luokkajärjestelmän tyypillisimmistä elementeistä on luokkien xtype-ominaisuus. Xtype-nimi on lyhyempi ja helpompi muistaa ja käyttää kuin luokan nimi ja se helpottaakin ohjelmistokehittäjän työtä. Xtype-ominaisuutta voidaan käyttää Ext.create()-funktion sijaan luodessa uusia instansseja luokista. Esimerkiksi äsken luotuun kontaktilistaan saadaan helposti lisättyä yläpalkki, jossa on otsikko ”Yhteystiedot” ja yksi painike, xtype-ominaisuuksien avulla seuraavasti:

```
Ext.define('HelloWorld.view.ContactList', {
      extend: 'Ext.dataview.List',
      xtype: 'contactlist',
      config: {
```

```
        items: [{
                    xtype: 'toolbar',
                    title: 'Yhteystiedot',
                    items: [{
                                xtype: 'spacer'
                                }, {
                                xtype: 'button',
                                iconCls: 'add',
                                ui: 'plain'
                    }]
        }]
        }
});
```

Sencha Touch sisältää runsaasti valmiita luokkia ja komponentteja, jotka on suunniteltu MVC-arkkitehtuurin periaatteiden mukaisesti. Sencha Touchin luokkakirjasto sisältää muun muassa malliluokkia, joiden avulla voidaan kuvata sovelluksessa liikkuvaa tietoa. Kommunikaatioluokat taas sisältävät määritykset sovelluksessa liikkuvan tiedon varastoinnista ja keskinäisestä viestinnästä. Tärkeimpiä kommunikaatioluokkia ovat Store-, Proxy- ja Reader-luokat. Näkymäluokat kuvaavat sovelluksen käyttöliittymää ja ulkoasua ja ovat sovelluksen näkyvin osa. Ohjainluokat toimivat liiman tavoin näkymien ja mallien välillä yhdistäen nämä toisiinsa ja sisältävät sovelluslogiikan kannalta tärkeimmät komennot. (Kosmaczewski 2013, 29.)

5.3 Sencha Touchin käyttöönotto

Toimiakseen Sencha Touch vaatii webkit-selaimen kuten Google Chromen tai Safarin. Mobiililaitteilla myös Internet Explorerin versiot 10 ja 11 tukevat Sencha Touchin uusimpia versioita. Oikean selaimen lisäksi täytyy asentaa Sencha Cmd. Sencha Cmd on Sencha-sovellusten päärakennustyökalu ja sen saa ladattua ilmaiseksi Senchan virallisilta internetsivuilta. Koska Sencha Cmd on kirjoitettu Java-ohjelmointikielellä, se vaatii toimiakseen JRE:n (Java Runtime Environment) version 1.7. (Sencha Docs 2014.)

Sencha Touch tarvitsee myös Rubyn luomaan Sencha Touchin käyttämän käännetyn CSS-tyylitiedoston. Ruby on dynaaminen avoimen lähdekoodin ohjelmointikieli, joka keskittyy selkeyteen ja tuottavuuteen (Ruby Programming Language 2014). Kun kaikki vaadittavat ohjelmistot on asennettu, voidaan ladata itse Sencha Touch sen virallisilta verkkosivuilta. Pakatut tiedostot puretaan sovelluskehittäjän projekteille varat-

tuun työtilaan, joka on yleensä lokaalissa web-kehityksessä käytettävän web-serverin www-kansio. Sencha Cmd-työkalun asennuksen voi tarkistaa menemällä komentorivillä kansioon, jonne Sencha Touch purettiin ja ajamalla komennon ”sencha”. Tämä tulostaa käytössä olevan Sencha Cmd:n versionumeron ja tiedot. Mikäli Sencha Cmd on asennettu oikein, voidaan ensimmäinen Sencha Touch sovellus luoda seuraavasti:

```
sencha generate app HelloWorld ../hello
```

Jotta komento toimisi, täytyy sovelluskehittäjän olla kansiossa, johon Sencha Touch purettiin. Komento luo hello-projektikansion HelloWorld-nimiselle sovellukselle Sencha Touch -kansion ulkopuolelle. (Sencha Docs 2014.)

5.4 Sencha Touch ja PhoneGap

Sencha Touch on pääasiassa laaja käyttöliittymäkehys, joka voidaan yhdistää PhoneGapin kanssa mobiililaitteen natiivien ominaisuuksien hyödyntämiseksi. Kun Sencha Touch -sovellus on luotu käyttäen Sencha Cmd-ohjelmistoa, se voidaan laajentaa käsittämään myös PhoneGap seuraavalla komennolla:

```
sencha phonegap init com.example.hello HelloWorld
```

Komennon kaksi viimeistä parametria ovat vapaaehtoisia. Jotta komento toimisi, täytyy komentorivillä ensin navigoida Sencha Touch -sovelluksen hakemistoon. Komento lisää sovelluksen projektikansioon phonegap-hakemiston, joka sisältää PhoneGapin tarvitsemat tiedostot. (Gerbasi 2013.)

Sencha Touch -sovelluksen työnkulku ei muutu PhoneGapin liittämisen jälkeen juuri lainkaan. Sovellus voi käyttää PhoneGapin tarjoamia rajapintoja hyödyntääkseen laitteen natiiveja ominaisuuksia, mutta rakentuu muuten samalla tavalla kuin mikä tahansa Sencha Touch -sovellus. Sovellus voidaan paketoida jakelua varten joko Senchan omalla Sencha Mobile Packager -työkalulla, Apache Cordovalla tai PhoneGap Build -pilvipalvelussa. Phonegap.local.properties-tiedostoon voidaan määrittää kohdealustat ja valita pakkausmenetelmäksi PhoneGap Build seuraavasti:

```
phonegap.platform=android ios wp8
```

```
phonegap.build.remote=true

phonegap.build.remote.username={PhoneGap Build tunnus}

phonegap.build.remote.password={PhoneGap Build salasana}
```

Jos ei haluta käyttää PhoneGap Build -pilvipalvelua sovelluksen pakkaamiseen, on huolehdittava kunkin kohdealustan vaatimasta kehitysympäristöstä. (Boonstra 2013.)

6 jQuery Mobile

6.1 Yleistä

JQuery Mobile on yhden maailman tunnetuimman ja suosituimman JavaScript-kirjaston, jQueryn, liitännäinen. JQueryn ensimmäinen versio julkaistiin vuonna 2006 ja se on tällä hetkellä käytetyin JavaScript-kirjasto web-sivustoilla (Usage of JavaScript libraries for websites 2014). JQuerylla voidaan vaivattomasti toteuttaa muun muassa animaatioita tai ajax-kyselyjä sekä se helpottaa html-dokumentin eri elementtien käsittelyä. JQueryn syntaksi on tehty mahdollisimman yksinkertaiseksi ja selkeäksi, mikä on osaltaan vaikuttanut sen suosioon web-kehittäjien keskuudessa. (jQuery 2014.)

JQuery Mobile on käyttöliittymäkehys, jonka avulla voidaan toteuttaa web-sovelluksia mobiililaitteille vaivattomasti; se tukee muun muassa iOS-, Android- ja Windows Phone 7.5 ja 8 -laitteita (jQuery Mobile 1.4 Supported Platforms 2014). JQuery Mobilen ensimmäinen versio julkaistiin loppuvuodesta 2010 ja uusin vakaa versio julkaistiin helmikuussa 2014. JQuery Mobile -sovellukset rakentuvat jQuery-ytimen päälle joten ne vaativat jQueryn toimiakseen. (Ortiz 2011.)

6.2 jQuery Mobile -sovelluksen rakenne

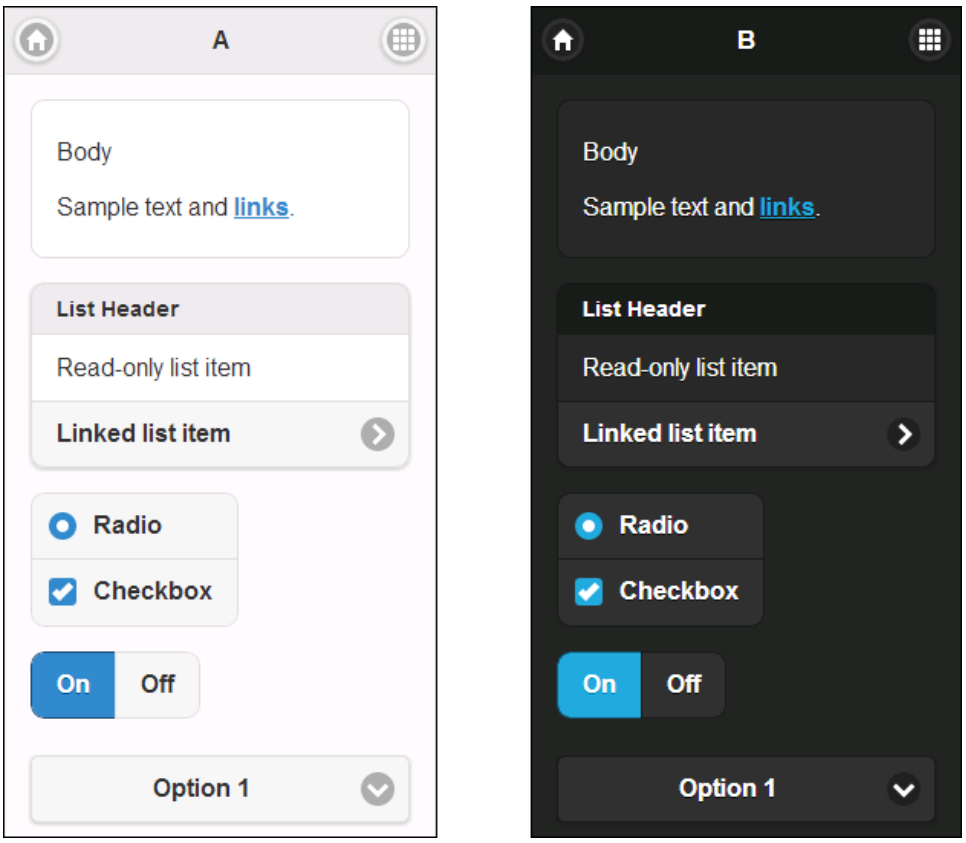
JQuery Mobile -sovellukset rakennetaan hyvin samaan tapaan kuin web-sivustot: sovellus sisältää tyylitiedoston, jQuery- ja jQuery Mobile -JavaScript-tiedostot sekä index.html-tiedoston, johon sovellus rakennetaan käyttöliittymäkomponenttien avulla. Sovelluksen eri näkymät luodaan div-elementteinä, joiden data role eli dataroolik-si asetetaan page eli sivu. Yksinkertaisen yksinäkymäisen jQuery Mobile -sovelluksen index.html voisi näyttää esimerkiksi tältä:

```
 <!doctype html>
<html>
<head>
    <title>Hei Maailma!</title>
    <meta name="viewport" content="width=device-width, initial-
scale=1">
```

```
    <link rel="stylesheet"
href="http://code.jquery.com/mobile/1.4.2/jquery.mobile-
1.4.2.min.css">
    <script src="http://code.jquery.com/jquery-
1.9.1.min.js"></script>
    <script
src="http://code.jquery.com/mobile/1.4.2/jquery.mobile-
1.4.2.min.js"></script>
</head>
<body>
    <div data-role="page">

        <div data-role="header">
            <h1>Ylätunniste</h1>
        </div>

        <div data-role="content">
            <p>Hei maailma!</p>
        </div>

        <div data-role="footer">
            <h4>Alatunniste</h4>
        </div>

    </div>
</body>
</html>
```

Sovellus on rakennettu samaan tapaan kuin html5-dokumentti: header-tagien sisällä sovellukseen linkitetään sekä tyylitiedosto että jQuery- ja jQuery Mobile -JavaScript-kirjastot. Body-tagien sisällä on itse sovelluksen käyttöliittymä: div-elementille on annettu datarooliksi sivu, joten se käyttäytyy sovellusnäkymän tavoin. Tälle näkymälle on määritetty ylätunniste eli header, sisältöosa ja alatunniste eli footer. Näkymän eri osioille on näkymän tapaan annettu myös dataroolit. (Colom, Magnuson & Sontag 2014.)

Koska jQuery Mobile on käyttöliittymäkehys, se sisältää runsaasti erilaisia valmiita käyttöliittymäkomponentteja (ks. kuvio 9, s. 37), joiden avulla sovelluksen rakentaminen sujuu vaivatta. Näihin komponentteihin lukeutuvat muun muassa painikkeet eli buttonit, paneelit, listanäkymät, popup- eli ponnahdusikkunat, lomake-elementit sekä erilaiset työkalupalkit.



Kuvio 9. jQuery Mobile -käyttöliittymäkehyksen käyttöliittymäkomponentteja kahdessa eri teemassa

Erilliset näkymät luodaan kaikki samaan index-html-tiedostoon div-elementteinä, joille annetaan datarooliksi page eli sivu. Navigointi eri näkymien väillä tapahtuu yksinkertaisimmin ankkurilinkkien avulla: painikkeen tai linkin osoitteeksi (href) asetetaan kohdenäkymän tunnus eli id ankkurilinkkinä seuraavasti:

```
<a href="#page2" data-role="button">Seuraava</a>
```

Yllä olevalla koodilla luodaan painike “Seuraava”, jota painamalla sovellus navigoi näkymään, jonka id:ksi on asetettu “page2”. Käyttöliittymäkomponenttien lisäksi jQuery Mobile sisältää useita eri metodeja. Esimerkiksi $.mobile.activePage-metodilla voidaan selvittää, missä näkymässä käyttäjä juuri sillä hetkellä on ja $.mobile.changePage-metodin avulla voidaan navigoida sovelluksessa haluttuun näkymään ilman, että käyttäjän tarvitsee painaa linkkiä. (Ortiz 2011.)

6.3 jQuery Mobile -sovelluksen käyttöönotto

JQuery Mobilen käyttöönotto on kokeneelle web-kehittäjälle helppoa: jQuery Mobile -sovelluksessa tarvittavat tiedostot (tyylitiedosto ja JavaScript) voidaan ladata jQuery Mobilen virallisilta web-sivuilta. Näiden lisäksi tarvitaan vielä jQuery, jota ilman jQuery Mobilea ei voi käyttää. Myös jQueryn voi ladata sen virallisilta verkkosivuilta. Vaihtoehtoisesti sekä jQuery Mobile että jQuery voidaan hakea CDN:n (content distribution network) eli sisällönjakeluverkoston kautta, jolloin tiedostoja ei tarvitse ladata tietokoneelle. JQuery Mobilen käyttöönottoa varten riittää siis yksi index.html-tiedosto, jonka head-tagien välissä ilmoitetaan jQuery Mobilen tyyli- ja JavaScript-tiedostojen sekä jQueryn sijainnit. (Ghatol & Patel 2012, 128-134.)

6.4 jQuery Mobile ja PhoneGap

JQuery Mobile sopii hyvin käytettäväksi yhdessä PhoneGapin kanssa: jQuery Mobilella toteutetaan sovelluksen käyttöliittymä ja PhoneGapin avulla saadaan hyödynnettyä mobiililaitteen natiiveja ominaisuuksia ja pakattua se sovelluskauppajakelua varten. JQuery Mobilen kolme vaatimaa tiedostoa sijoitetaan PhoneGap-sovelluksen www-kansioon seuraavasti: tyylitiedosto css-kansioon ja jQuery Mobile ja jQuery -JavaScript-tiedostot js-kansioon. Tiedostot voidaan hakea sovellukseen myös CDN-sisällönjakeluverkoston kautta. Näiden tiedostojen sijainnit määritellään lopuksi sovelluksen index.html-tiedoston head-osioon. (Mts. 128-134.)

7 Aarrejahti-sovellus

7.1 Lähtökohdat ja sovelluksen kuvaus

Usealla jyväskyläläisen mainostoimiston, MEOM Oy:n, asiakkaalla on ollut käynnissä erilaisia aarteenetsintäkampanjoita. Näissä kampanjoissa kampanjan järjestäjä on piilottanut Jyväskylän alueelle ”aarteen”, joka aarteenetsintäkilpailun osallistujien on tarkoitus löytää. Sosiaalisessa mediassa, esimerkiksi kilpailun järjestäjän Facebook-sivulla, on jaettu vihjekuva, jonka perusteella kilpailijat ovat lähteneet etsimään aarretta. Kilpailijoiden etsimät aarteet ovat olleet joko julisteita tai tarroja, joissa oleva koodi aarteen löytäneen kilpailijan on käytävä syöttämässä kampanjalle tehdyn web-sivun lomakkeeseen. Lomakkeeseen syötetty aarteen koodi tarkistetaan ja mikäli koodi on oikein ja palkintoa ei ole kukaan muu kilpailija vielä voittanut, kilpailuun osallistunut täyttää lomakkeeseen myös yhteystietonsa palkinnon lähettämistä varten.

Tässä opinnäytetyössä toteutettiin hybridisovellus, joka toimisi aarteenetsijän apuna aarrejahtikilpailussa. Sovellus sisältäisi kaikki sillä hetkellä käynnissä olevat aarteenetsintäkampanjat ja näiden aarteenetsinnät. Aarteenetsinnät jaettiin kolmeen eri vaiheeseen: ensimmäisessä vaiheessa käyttäjälle näytetään ensimmäinen vihjekuva, joka on suurpiirteinen ja antaa vihjeen siitä, mistä päin Jyväskylää aarretta kannattaa lähteä etsimään. Tämän lisäksi sovellus kertoo, lähestyykö kilpailija aarretta vai kulkeeko hän siitä poispäin. Kun kilpailija on tarpeeksi lähellä aarteen sijaintia, hän pääsee siirtymään toiseen vaiheeseen, jossa hänelle näytetään toinen edellistä tarkempi vihjekuva. Kun aarre löytyy, kilpailija siirtyy kolmanteen vaiheeseen, jossa hän syöttää sovellukseen aarteen koodin. Mikäli koodi on oikein, pääsee käyttäjä syöttämään yhteystietonsa sovelluksessa olevaan lomakkeeseen palkinnon lähettämistä varten.

7.2 Sovelluksen toteutus

7.2.1 Sovelluksen vaatimukset

Sovellus tuli toteuttaa hybridisovelluksena ja siinä oli tarkoitus hyödyntää hybridisovelluskehityksen parhaimpia puolia: sovelluskehityksen tulisi olla mahdollisimman vaivatonta ja samaa sovelluskoodia täytyisi kyetä hyödyntämään kaikilla kohdealustoilla. Sovelluksen tuli toimia Suomen suosituimmilla mobiililaitteiden käyttöjärjestelmillä: Androidilla, iOS:lla ja Windows Phonella. Näiden lisäksi sovelluksen täytyi pystyä paikallistamaan käyttäjän sijainti ja lähettää voittajan yhteystiedot kampanjan järjestäjän sähköpostiin. Sovellusta käytetään pääasiallisesti älypuhelimilla, joten tablet-laitteita ei erikseen huomioitu.

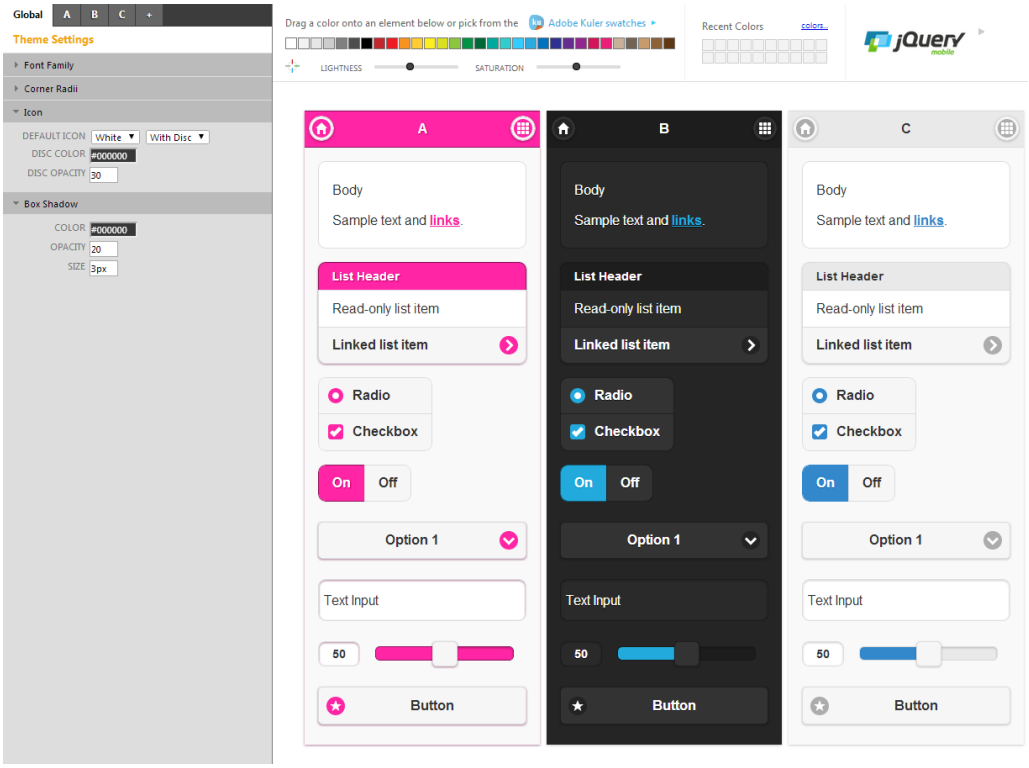
7.2.2 Käytettävien teknologioiden valinta

Hybridisovelluksen toteutukseen valittiin PhoneGap-ohjelmistokehys, jonka sijainti- eli geolocation-rajapintaa käytettiin käyttäjän sijainnin määrittämiseen. PhoneGap-sovelluksen kehittämiseen on olemassa kaksi eri lähestymistapaa: PhoneGapin oma sisäänrakennettu buildaus eli paketointi, jolloin jokaiselle eri kohdekäyttöjärjestelmälle täytyy olla niiden vaatimat sovelluskehitysympäristönsä (ks. taulukko 2, s. 12), ja PhoneGap Build -pilvipalvelun käyttäminen sovelluksen paketoimiseen, jolloin sovellus kirjoitetaan kerran ja paketoidaan pilvipalvelussa kaikille kohdealustoille. Jotta sovelluskehitys olisi mahdollisimman vaivatonta, Aarrejahti-sovelluksen toteutukseen päätettiin käyttää Phonegap Build -palvelua sovelluksen paketointiin Androidille, iOS:lle ja Windows Phone 8:lle. Opinnäytetyössä tutustuttiin myös kahteen web-sovellusten toteuttamiseen tarkoitettuun käyttöliittymäkehykseen: Sencha Touchiin ja jQuery Mobileen, joista jQuery Mobile valittiin sovelluksen käyttöliittymän toteutukseen.

Sencha Touch ja jQuery Mobile eroavat toisistaan monin tavoin: Sencha Touch sisältää käyttöliittymäkomponenttien lisäksi oman sisäänrakennetun MVC-arkkitehtuurinsa, jota jQuery Mobilesta ei valmiina löydy. Halutessaan sovelluskehittäjä voi yhdistää jQuery Mobileen jonkin MVC-arkkitehtuurin sisältävän sovelluskehyksen, kuten Backbone.js:n tai Angular.js:n. Kokeneelle jQuery-kehittäjälle jQuery Mobilen omaksuminen tulee lähes luonnostaan kun taas Sencha Touchin luokkajär-

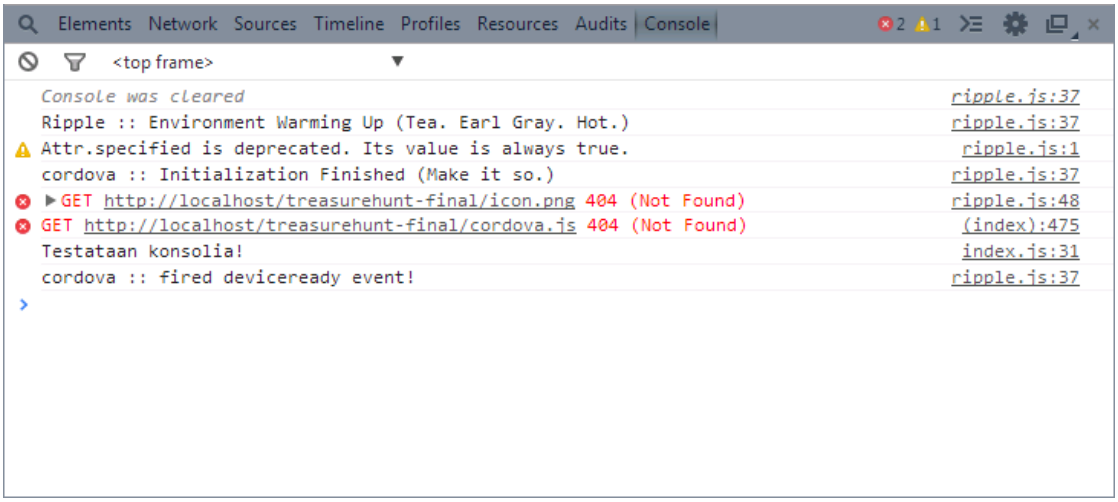
jestelmä ja täysin JavaScriptillä kirjoitettu sovelluslogiikka voivat olla hankalia sisäistää. JQueryn suuren suosion myötä myös jQuery Mobile on saanut taaksensa paljon käyttäjiä ja vahvan yhteisön: mm. ohjelmistokehittäjien suosimalta Stackoverflow-sivustolta löytyy runsaasti tukea jQuery Mobile -sovelluksen tekoon. Sencha Touch -sovelluskehittäjän on huomattavasti vaikeampi löytää tukea ja apua ongelmiin Senchan oman virallisen sivuston ulkopuolelta. Senchan oma tukipalvelukin on maksullinen lisäpalvelu.

Sencha Touch -sovellukset ovat yleensä jQuery Mobile -sovelluksia sulavampia ja ne sekä tuntuvat että näyttävät enemmän natiivisovelluksilta kuin jQuery Mobilella tehdyt sovellukset. JQuery Mobilella tehdyt sovellukset usein näyttävät keskenään hyvin samankaltaisilta ja vaatisivat sovelluskehittäjältä myös osaamista graafisesta suunnittelusta. JQuery Mobilelle on kuitenkin kehitetty Themroller-työkalu (ks. kuvio 10), joka mahdollistaa oman jQuery Mobile -teeman luomisen ilman css-osaamista. Themeroller-työkalulla voidaan jQuery Mobile -teemaan luoda oma swatch- eli värimalli, jolle teeman luoja asettaa fontit, värit, linkkien muotoilut sekä reunojen pyöritystykset. Luotu teema ladataan Themerollerista zip-pakettina, joka sisältää teemasa käytettävät tyylitiedostot sekä kuvakkeet.



Kuvio 10. Themeroller-työkalun käyttöliittymä

Sencha Touchissa teemat luodaan SASS-kielellä. SASS on CSS-preprosessori, joka tuo CSS:n perussyntaksiin lisäominaisuuksia kuten muuttujia, matemaattisia funktioita sekä periytyviä tyylimäärittelyjä. SASS täytyy kääntää CSS:ksi ennen kuin sitä voidaan käyttää. JQuery Mobilessa virheenkorjaus eli debuggaus on helppoa: sovellusta voidaan testata tietokoneella selaimessa, jossa on sovelluskehitykseen tarkoitetut työkalut asennettuina. Esimerkiksi Google Chromella JavaScript ja html-virheet voidaan nähdä valitsemalla selaimen valikosta työkalut ja kehittäjän työkalut tai pikanäppäinyhdistelmällä ctrl + shift + I. Kehittäjän työkalut saa näkyviin myös klikkaamalla html-dokumenttia hiiren oikealla painikkeella ja valitsemalla ”tarkastele elementtiä”. Konsoli-välilehdeltä (ks. kuvio 11) voidaan tarkastella mahdollisia JavaScript-virheitä ja millä rivillä koodissa ne mahdollisesti ilmenevät. Mozilla Firefoxille on saatavilla vastaavanlainen Firebug-työkalu.



Kuvio 11. Google Chrome -selaimen kehittäjän työkalut

Sencha Touchin debuggaaminen on jQuery Mobileen verrattuna vaikeaa: jos koodissa on virhe, se ei välttämättä näy kehittäjän työkalujen konsolissa ollenkaan. Pienetkin virheet Sencha Touch -sovelluksen koodissa voivat aiheuttaa koko sovelluksen toimimattomuuden ja kokemattoman JavaScript-kehittäjän on vaikea löytää virhettä. Sencha Touchin dokumentaatio puolestaan on hyvinkin laaja ja kattava: käyttöliittymäkomponenttien lisäksi dokumentaatio käy läpi muun muassa metodit, luokkajärjestelmän, datan käsittelyn ja näkymien luonnin. JQuery Mobilen dokumentaatio

keskittyy lähinnä esittämään esimerkkejä käyttöliittymäkomponenteista ja muu osuus jää lähes käsittelemättä.

Molemmilla frameworkeilla on omat vahvuutensa: Sencha Touch sopii paremmin laajempiin ja monimutkaisiin sovelluksiin sisäänrakennetun MVC-arkkitehtuurin ja sulavamman käyttöliittymän vuoksi. JQuery Mobile taas on hyvä työkalu kokeneille jQuery-kehittäjille hieman yksinkertaisimpiin perussovelluksiin. Taulukkoon 4 on koottu molempien vahvuudet (vihreä) sekä heikkoudet (punainen). Aarrejahtisovelluksen tekoon valittiin jQuery Mobile sen nopean käyttöönoton ja helpon debuggauksen vuoksi. Sovellus on toiminnaltaan myös melko yksinkertainen, joten jQuery Mobile sopi sovelluksen toteuttamiseen hyvin.

Taulukko 4. Sencha Touch ja jQuery Mobile -käyttöliittymäkehyksien vertailu

Sencha Touch	jQuery Mobile
Nopea & sulava	Helppo & nopea käyttöönotto
Tuntuu ja näyttää natiivilta	Paljon ulkopuolista tukea (mm. Stackoverflow)
MVC-arkkitehtuuri	Themeroller-teemat
Kattava & laaja dokumentaatio	Helppo virheiden debuggaus
Monimutkainen → hidas käyttöönotto	Voi olla melko hidas
Ei juurikaan apua virallisten sivujen ulkopuolelta, tukipalvelu maksullinen	Sovellukset näyttävät kaikki usein samanlaisilta
Virheiden debuggaus vaikeaa	Dokumentaatio vaillinainen

7.3 Arkkitehtuuri

Aarrejahti-sovellus on arkkitehtuurillisesti melko yksinkertainen. Koska sovelluksen paketointiin käytetään PhoneGap Build -pilvipalvelua, ei PhoneGapista tarvita kuin www-kansio ja config.xml-tiedosto, johon määritellään sovelluksen asetuksia PhoneGap Buildin pakkausta varten. Koska sovelluksen käyttöliittymän toteutukseen valittiin jQuery Mobile, tarvittiin sekä jQueryn että jQuery Mobilen vaatimat tiedostot. Sovelluslogiikka on kirjoitettu yhteen JavaScript-tiedostoon ja käyttöliittymä rakennettu index.html-tiedostoon. Sovelluksen vaatimat kunkin eri kohdealustan kuvak-

keet ja aloitusnäyttökuvat (splashscreen) sijoitettiin res-kansioon. Sovelluksen saamiseksi helposti PhoneGap Buildiin sovelluksessa käytettiin Git-versionhallinan GitHub-työkalua, joka lisää sovelluskansion juureen omat tiedostonsa. Koko sovelluksen kansiorakenne on kuvattu kuviossa 12.

Kuvio 12. Aarrejahti-sovelluksen kansiorakenne

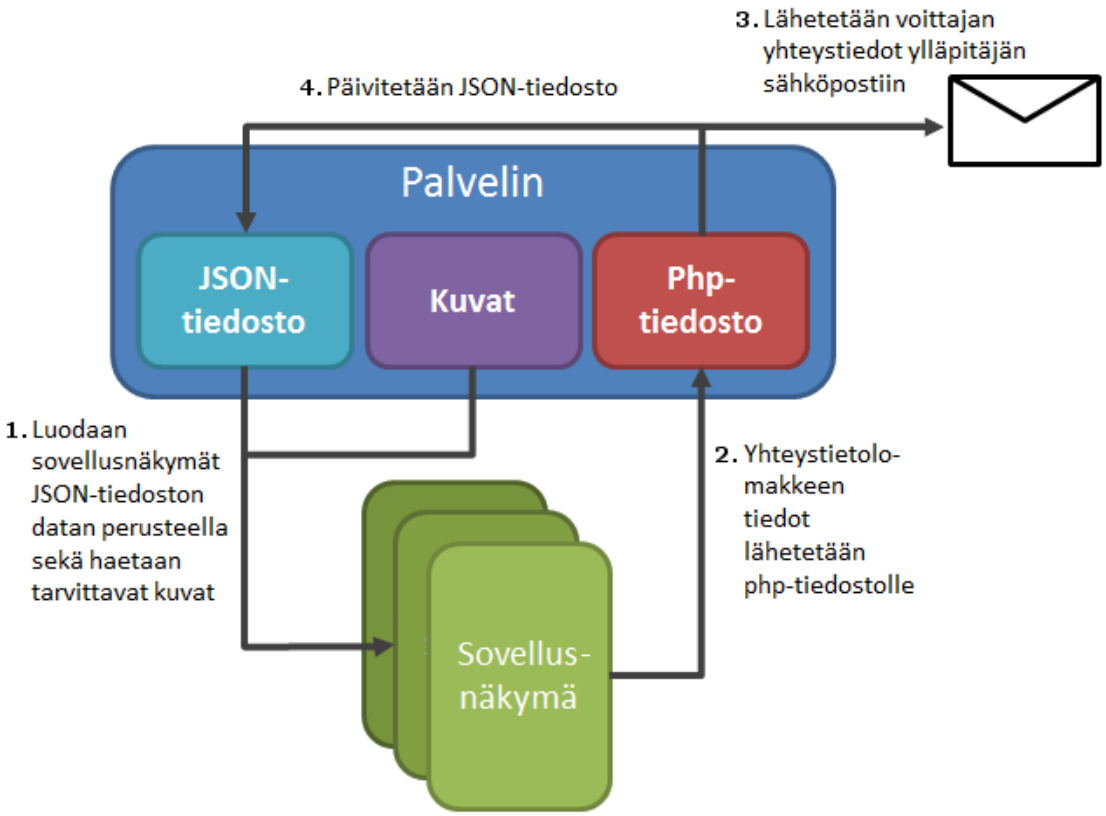
Itse sovellus koostuu kampanjoista ja niihin kuuluvista aarrejahdeista. Sekä kampanjat että aarrejahdit haetaan sovellukseen JSON-tiedostosta palvelimelta. JSON (JavaScript Object Notation) on muun muassa JavaScript-ohjelmissa käytettävä yksinkertainen tiedonsiirtomuoto, jota käytetään nykyään erittäin laajasti kaikessa asiakkaan ja palvelimen välisessä liikenteessä. Aarrejahti-sovelluksessa käytettävä JSON-tiedosto on rakenteeltaan seuraavanlainen:

```
{
    "campaigns": [
        {
            "campaignID": "campaign01",
```

```
                "campaignName": "Kampanja1",
                "campaignLogo": "kampanjan1logonosoite.jpg",
                "campaignDate": "22.03.2014",
                "campaignDescription": "Kampanja1:n kuvaus.",
                "treasurehunts": [
                    {
                     "treasurehuntID": "campaign01treasurehunt01",
               "treasurehuntName": "Aarrejahti1",
               "treasurehuntLat": "62.24162",
               "treasurehuntLon": "25.75973",
               "treasurehuntStatus": "0",
               "treasurehuntCode": "KAMPANJAN1AARREJAHDIN1KOODI",
               "treasurehuntStep1": "http://vihjekuvan1osoite.jpg",
               "treasurehuntStep2": "http://vihjekuvan2osoite.jpg"
                    }
                ]
            }
        ]
    }
```


JSON-tiedostossa campaigns-sisältää kaikki sovelluksen kampanjat. Kampanjoilla on aina tunnus eli ID, joka on muotoa campaignXX, jossa XX on kampanjan numero. Tunnuksen lisäksi kampanjoille asetetaan nimi, logo-kuvan osoite, kampanjan lisäyspäivämäärä, kuvaus ja kaikki kampanjaan kuuluvat aarrejahdit. Aarrejahdeilla on myös tunnus, joka on aina muotoa campaignXXtreasurehuntXX, jossa campaignXX on sen kampanjan tunnus, johon ko. aarrejahti kuuluu ja trasurehuntXX on aarrejahdin oma uniikki tunnus. Aarrejahdeille annetaan nimi, aarteen sijainnin leveys- ja pituusasteet (latitude ja longitude), aarrejahdin tila eli status (0 tarkoittaa käynnissä olevaa aarrejahtia ja 1 suljettua), aarteen koodi sekä kahden vihjekuvan osoitteet.

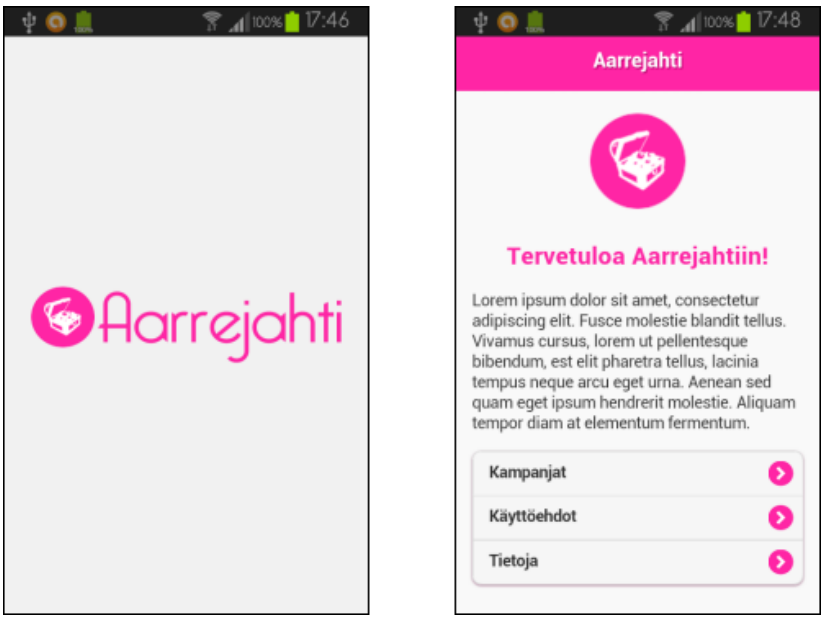
Sovellus luo kampanjoille ja aarrejahdeille omat näkymänsä sovellukseen JSON-tiedostosta poimitun datan perusteella. Aarrejahdit koostuvat kolmesti eri vaiheesta: 1. vaiheessa aarteenetsijälle näytetään 1. vihjekuva, toisessa vaiheessa 2. vihjekuva ja 3. vaiheessa aarteenetsijä syöttää ensin sovellukseen aarteen koodin, ja mikäli koodi on oikein, hän pääsee täyttämään yhteystietonsa palkinnon lähettämistä varten. Yhteystietolomakkeen tietojen lähettäminen hoidetaan palvelimella olevalla php-tiedostolla, joka lähettää yhteystiedot kampanjan ylläpitäjän sähköpostiin sekä muuttaa ko. aarrejahdin tilan suljetuksi JSON-tiedostoon. Koko sovellusarkkitehtuuri on kuvattu kuviossa 13 (s. 46).



Kuvio 13. Aarrejahti-sovelluksen arkkitehtuuri

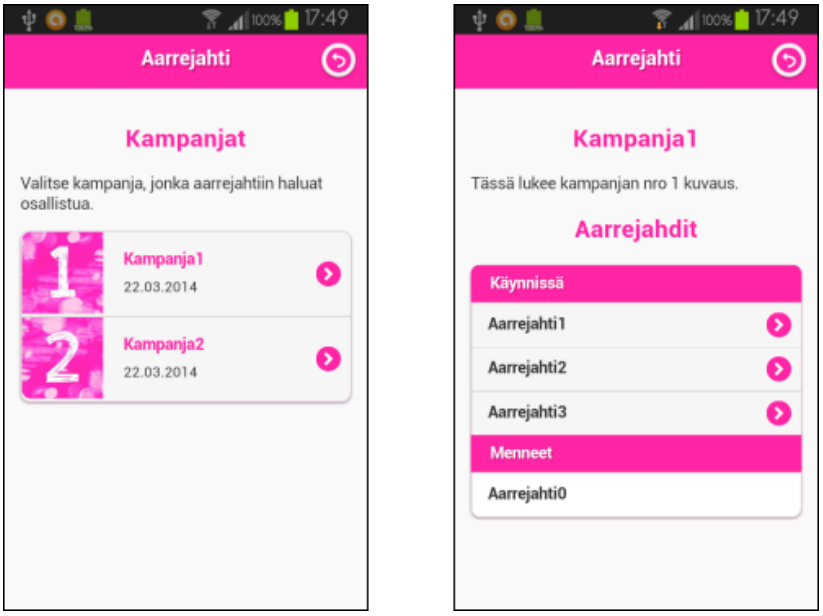
7.4 Sovelluksen esittely

Sovelluksen käynnistyessä käyttäjälle näytetään aloituskuva (splash screen). Aloituskuva haetaan käytettävän mobiililaitteen perusteella sovelluksen res-hakemiston screen-kansiosta. Aloituskuva näkyy muutaman sekunnin ajan, jonka jälkeen sovelluksen päänäkymä tulee esiin. Päänäkymässä on sovelluksen logo, lyhyt esittelyteksti ja mahdollisuus jatkaa tarkastelemaan käynnissä olevia kampanjoita, sovelluksen käyttöehtoja tai tietoja (ks. kuvio 14, s. 47). Sovelluksen jokaisessa näkymässä on ylätunniste eli header, jossa ”Aarrejahti”-tekstistä pääsee sovelluksen päänäkymään ja alanäkymissä headerin oikessa reunassa olevasta painikkeesta pääsee yhden näkymän takaisinpäin.



Kuvio 14. Aarrejahti-sovelluksen aloituskuva ja päänäkymä

Käyttöehdot- ja Tietoja-näkymät ovat rakenteeltaan keskenään samanlaisia: ne sisältävät muiden näkymien tapaan ylätunnisteen ja sisältöosan. Ylätunnisteen oikeassa reunassa näissä näkymissä on takaisin-painike. Kampanjat-näkymä sisältää kaikki sillä hetkellä käynnissä olevat kampanjat. Kampanjoista esitetään tässä näkymässä nimi, logo ja kampanjan käynnistymispäivämäärä. Yksittäisen kampanjan sivulla on lyhyt kuvaus itse kampanjasta, sen palkinnoista, osallistumisajasta jne, sekä lueteltuna kaikki kampanjaan kuuluvat aarrejahdit, jotka erotellaan käynnissä oleviin ja menneisiin (ks. kuvio 15).



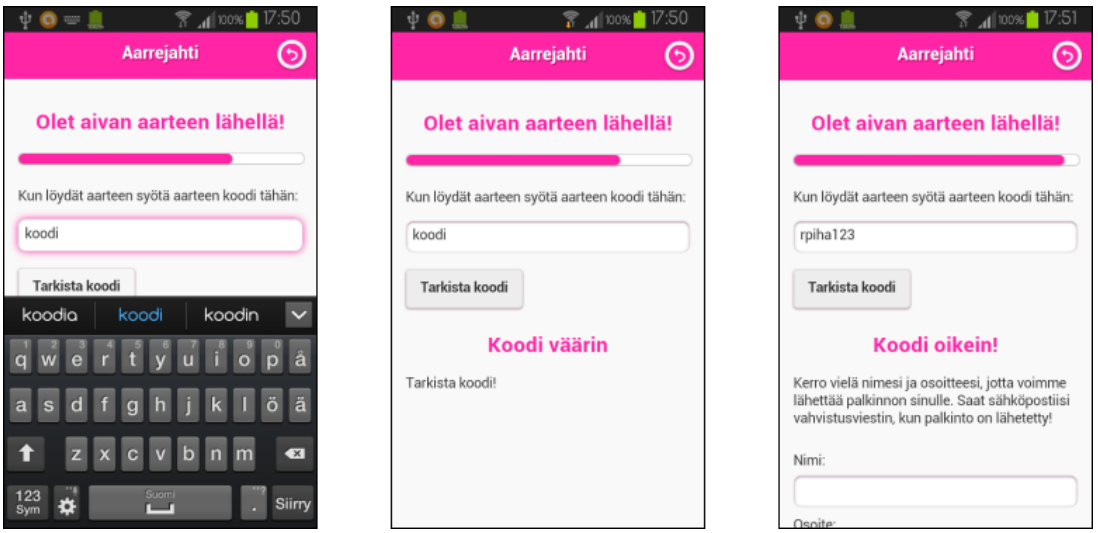
Kuvio 15. Aarrejahti-sovelluksen käynnissä olevien kampanjoiden ja yksittäisen kampanjan näkymät

Yksittäinen aarrejahti alkaa aarrejahdin 1. vaiheella, jossa aarteenetsijälle näytetään ensimmäinen vihjekuva (ks. kuvio 16). Vihjekuvan yllä on aarrejahdin etenemistä kuvaava edistymispalkki (progress bar), joka täyttyy sitä mukaa kun aarteenetsijä lähenee aarteen sijaintia, sekä otsikko, jossa oletuksena lukee ”Etsitään sijaintia”. Kun paikallistaminen onnistuu, otsikon sisältö riippuu aarteenetsijän sijainnista aarteeseen nähden: ”Et ole lähellä aarretta, etsiskelehän vielä”, ” Olet vielä melko kaukana aarteesta”, ”Olet melko lähellä aarretta” ja ”Olet lähellä aarretta!”. Vihjekuvan alla on myös edistymisestä tiedottava vihje, joka aarteenetsijän sijainnista riippuen on joko ”kylmenee”, ”lämpenee”, ”ei kylmene eikä lämpene” tai ”polttaa”. Aarteenetsijällä on mahdollisuus tarkistuttaa sijaintinsa vihjekuvan alta löytyvästä painikkeesta. Oletuksena sovellus tarkistaa käyttäjän sijainnin 5 sekunnin välein ja päivittää tarvittaessa vihjeet ja edistymispalkin. Kun aarteenetsijä on tarpeeksi lähellä aarteen sijaintia, vihjekuvan alle ilmestyy painike, josta pääsee seuraavaan vaiheeseen (ks. kuvio 16 oikealla).

Kuvio 16. Aarrejahti-sovelluksen yksittäisen aarrejahdin 1. vaihe

Aarrejahdin 2. vaiheen käyttöliittymä on täysin identtinen 1. vaiheen kanssa: vain vihjekuva on eri. Toisen vihjekuvan on tarkoitus olla tarkempi kuin ensimmäisen, joka on suurpiirteisempi vihje siitä alueesta Jyväskylää, jolle aarre on kätketty. 2. vaiheessa on samat vihjeet ja edistymispalkki kuin 1. vaiheessakin ja kun aarteenetsijä on aivan aarteen lähettyvillä, pääsee hän siirtymään kolmanteen ja viimeiseen vaiheeseen.

3. vaiheessa käyttäjälle ilmoitetaan, että hän on aivan aarteen lähettyvillä. Kun aarteenetsijä on paikallistanut aarteen, hän syöttää aarteen koodin sovelluksessa olevaan kenttään. Koodin pienillä ja isoilla kirjaimilla ei ole merkitystä vaan kaikki kenttään syötetyt kirjaimet muunnetaan pieniksi kirjaimiksi ennen vertailua oikean koodin kanssa. Mikäli koodi on väärin, näytetään käyttäjälle ilmoitus koodin virheellisyydestä (ks. kuvio 17 keskellä). Koodin ollessa oikein avautuu koodikentän alle yhteystietolomake, jonka aarteenetsijä täyttää (ks. kuvio 17 oikealla). Mikäli lomakkeen lähetys onnistuu, sen tiedot lähetettään kampanjan ylläpitäjälle ja JSON-tiedostoon päivitetään aarrejahdin tila. Sovellukseen tulee tieto onnistuneesta lähetyksestä ja siitä, että voittajaan otetaan piakkoin yhteyttä. Mikäli aarrejahdilla on jo voittaja, käyttäjälle ilmoitetaan, että palkinto on jo voitettu.



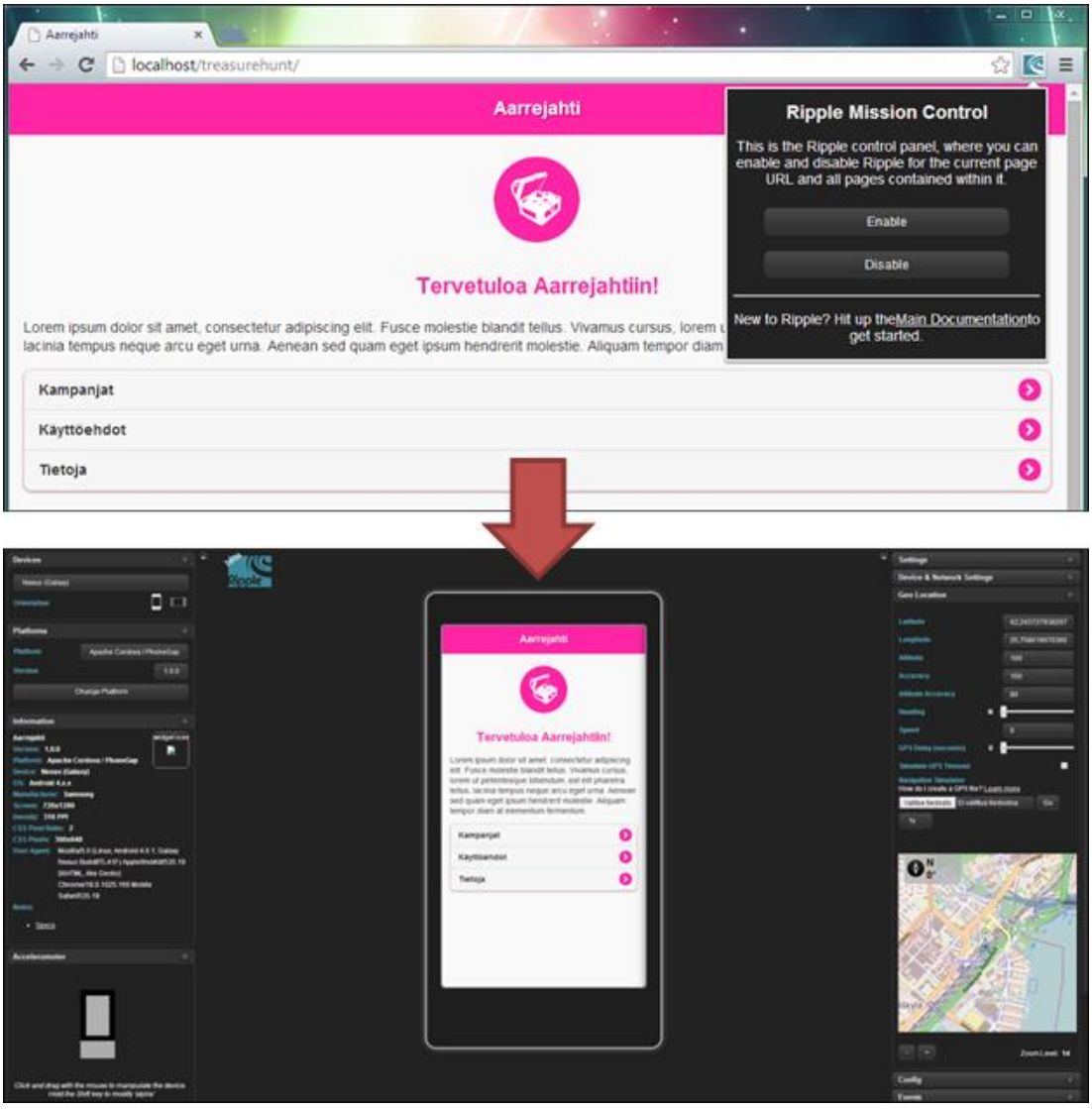
Kuvio 17. Aarrejahti-sovelluksen yksittäisen aarrejahdin 3. vaihe

7.5 Sovelluksen testaus

7.5.1 Sovelluksen testaus Ripple-PhoneGap-emulaattorilla

Aarrejahti-sovellusta testattiin Ripple-PhoneGap-emulaattorilla sen toteutuksen aikana. Apache Ripple on web-pohjainen mobiiliympäristösimulaattori, joka on suunniteltu web-mobiilisovellusten ripeään kehitykseen muun muassa Apache Cordova- ja BlackBerry WebWorks -ohjelmistokehyksille. Ripplen lisäksi PhoneGap-emulaattori vaatii toimiakseen Google Chrome -selaimen, johon Ripple asennetaan liitännäisenä.

Ripple-liitännäinen lisää Google Chromen selainikkunan oikeaan yläreunaan kuvakkeen, josta emulaattorin voi kytkeä päälle ja pois (ks. kuvio 18). Mitä tahansa websivustoa voi ajaa Ripple-emulaattorissa: kytkee vain emulaattorin päälle Ripplen kuvakkeesta. Aarrejahti-sovelluksen sovelluskehitysympäristön web-palvelimena käytettiin Wamp Server 2.4 -palvelinta ja sovellus rakennettiin palvelimen www-kansioon, jolloin sitä voitiin ajaa selaimessa osoitteessa localhost/treasurehunt/.



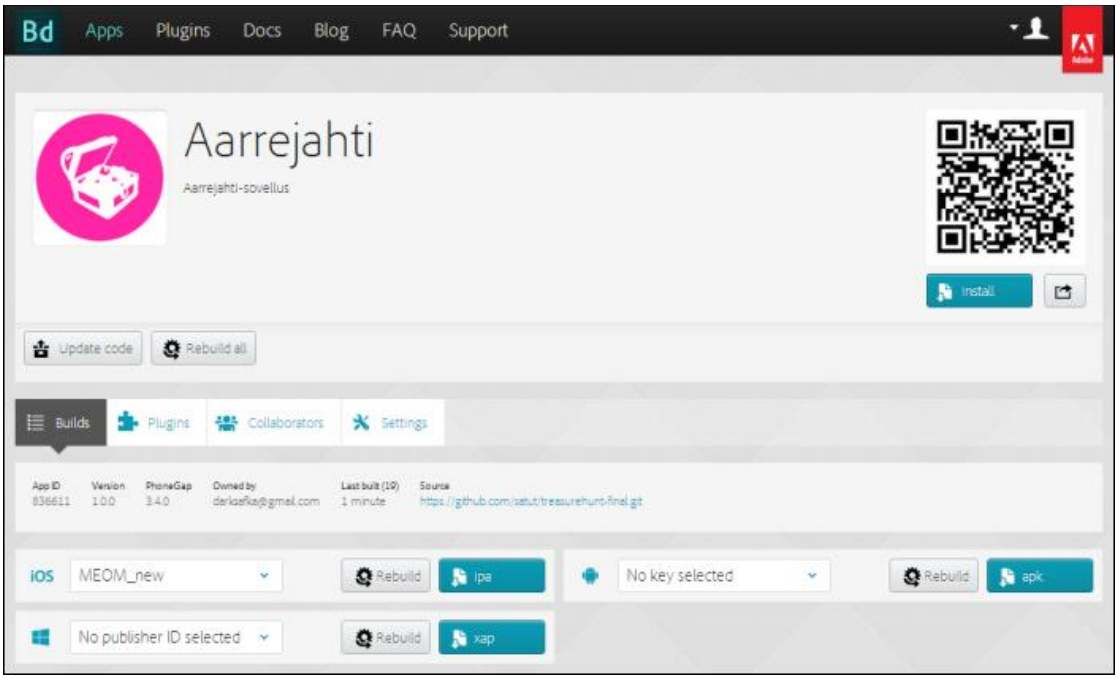
Kuvio 18. Ripple-PhoneGap-emulaattorin kytkeminen päälle Google Chrome -selaimessa

Ripple-emulaattori tukee osaa PhoneGapin ohjelmointirajapinnoista, joten sillä voidaan hyvin demota esimerkiksi sijainnin määrittämistä. Oman sijainnin voi emulaattorin määrittää joko tarkkoina leveys- ja pituusasteina tai oikeassa laidassa olevan kartan avulla. Aarteenetsinnän edistymistä pystyttiin simuloimaan siirtelemällä kartan osoitinta lähemmäs tai loitommas aarteen sijaintiin nähden. Emulaattorissa on

myös mahdollisuus valita kohdealusta pienestä määrästä erilaisia mobiililaitteita. Kuviossa 18 (s. 50) laitteeksi on valittu Samsung Galaxy Nexus -älypuhelin. Sovellusalueen koko ja muoto muuttuvat valitun kohdealustan mukaan.

7.5.2 Sovelluksen testaus älypuhelimilla

Jotta PhoneGap Build -sovellusta voitaisiin testata mobiililaitteella, täytyy se ensin pakata eli buildata. Pakkaamista varten sovellustiedostot täytyy ladata PhoneGap Buildiin joko zip-pakettina tai vaihtoehtoisesti hakea sovelluskehittäjän GitHub-talletuspaikasta, kuten Aarrejahti-sovelluksen tapauksessa tehtiin. Tällöin PhoneGap Buildiin kirjaudutaan sisään GitHubin tunnuksilla, sovellustyypiksi valitaan avoimen lähdekoodin (open source) sovellus ja sovelluksen pakattavien tiedostojen hakemisto valitaan sovelluskehittäjän GitHub-tunnuksen alla olevista talletuspaikoista. Kun talletuspaikka on valittu PhoneGap Build pakkaa sovelluksen kaikille niille kohdealustoille, jotka on määritelty sovelluksen config.xml-tiedostoon (ks. kuvio 19). iOS-sovellukset vaativat sovelluksen pakkausta varten iOS-allekirjoitusavaimen, joka ladataan erikseen PhoneGap Buildiin.



Kuvio 19. Aarrejahti-sovellus PhoneGap Buildissa

Android-laitteella voi paketoitua sovellusta testata lukemalla PhoneGap Buildista sovelluksen QR-koodin (ks. kuvio 19) tai vaihtoehtoisesti avaamalla laitteen internet-selaimella sovelluksen latausosoitteen. Molemmissa tapauksissa Android-mobiililaite

kysyy ensin, halutaanko sovellus asentaa ja kerrotaan, mitä käyttöoikeuksia sovellus tarvitsee. Kaikki PhoneGap-sovellukset vaativat vähintään täydet verkkoviestintäoikeudet ja tämän lisäksi sovellus tarvitsee oikeudet kutakin eri sovelluksessa käytettyä ohjelmointirajapintaa varten: Aarrejahti-sovelluksen tapauksessa verkkoviestinnän lisäksi sovellus tarvitsee oikeuden käyttäjän sijainnin määrittämiseen. Mikäli käyttäjä sallii sovelluksen vaatimukset, sovellus asennetaan laitteeseen.

iOS-mobiililaite vaatii PhoneGap Buildissa tapahtuvaa pakkausta varten iOS-allekirjoitusavaimen, joka ladataan PhoneGap Buildiin sovelluksen mukaan. iOS-allekirjoitusavainta varten tarvitaan joko Applen sovelluskehittäjätunnus tai rekisteröityminen Applen sovelluskehitysohjelmaan. iOS-allekirjoitusavain täytyy ensin muuntaa P12-tiedostoksi ennen kuin se voidaan ladata PhoneGap Buildiin: tämän voi tehdä joko Mac- tai Windows-tietokoneella. Mac-koneella muunto tapahtuu Xcode-ohjelmointiympäristössä, kun taas Windows-koneeseen täytyy asentaa OpenSSL, joka on avoimen lähdekoodin SSL- (Secure Sockets Layer) ja TLS- (Transport Layer Security) protokollatoteutus sekä salauskirjasto. Sekä P12-tiedostoksi muunnettu allekirjoitusavain että testattavalle laitteelle Applen sovelluskehityssivulla luotu varusteltu profiili (provisioning profile) ladataan PhoneGap Buildiin iOS-sovelluksen kirjautumisavaimeksi. Tämän jälkeen sovellus pakataan, jonka jälkeen se voidaan asentaa laitteeseen joko lukemalla QR-koodin tai avaamalla sovelluksen latauslinkin.

Windows Phone 8 -älypuhelimet täytyy ennen testausta rekisteröidä sovelluskehityskäyttöön. Tätä varten tarvitaan Windows Phone 8 SDK ja joko Microsoft-tunnus (ent. Windows Live) tai Windows Phone -kehittäjän tunnus. Windows Phone 8 SDK vaatii Windows 8 -käyttöjärjestelmän eikä sitä voi asentaa vanhemmille Windowsin versioille. SDK sisältää Windows Phone -laitteiden rekisteröintityökalun (Windows Phone Developer Registration Tool), jonka avulla puhelin rekisteröidään sovelluskehitystä varten. Kun puhelin on rekisteröity, voidaan PhoneGap Buildissa pakattu sovellus asentaa siihen Windows Phone 8 SDK:n sovellusten käyttöönottoon tarkoitetulla työkalulla (Application Deployment tool). Tätä varten puhelin on kytkettävä tietokoneeseen USB-kaapelilla.

Aarrejahti-sovelluksen kohdealustat olivat iPhone 4 ja sitä uudemmat IPhonet, Android 4.1.X ja sitä uudemmat Android-älypuhelimet ja Windows Phone 8 -älypuhelimet. Sovelluksen päätestauslaitteena toimi koko sovelluksen toteutuksen ajan Android 4.1.2 Jelly Bean -käyttöjärjestelmällä varustettu Samsung Galaxy S2 -älypuhelin, sillä Android-moiililaitteet eivät vaatineet testausta varten laitteen rekisteröimistä tai sovelluskehittäjän tunnistetietoja. Sovellus testattiin lopuksi myös Nokia Lumia 820 ja iPhone 4 -älypuhelimilla (ks. taulukko 5).

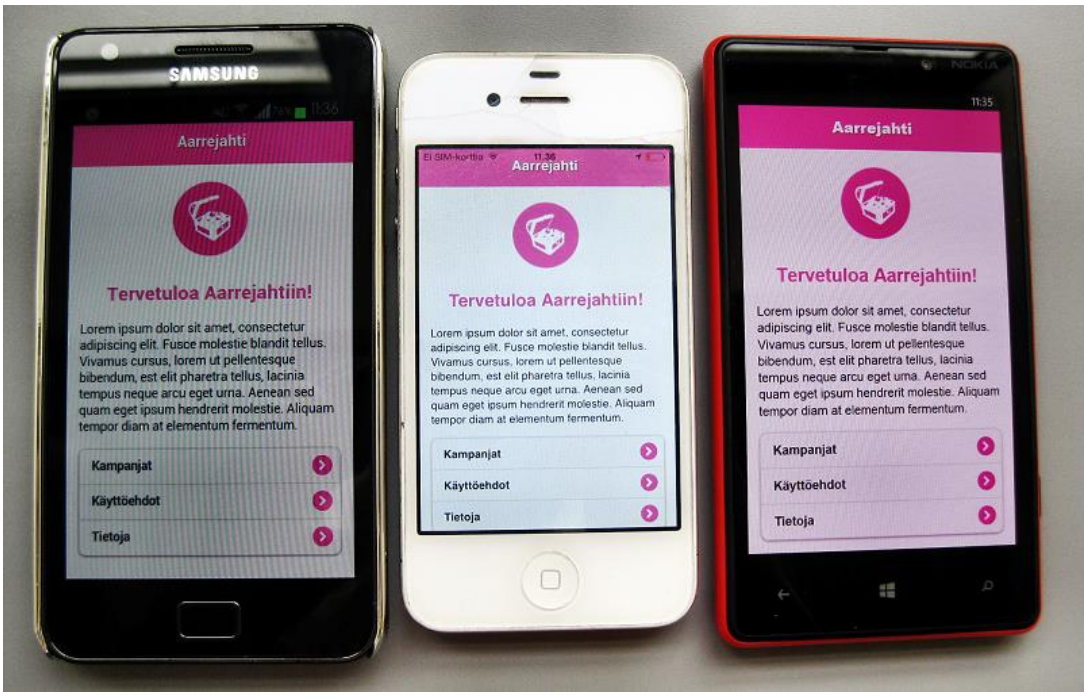
Taulukko 5. Aarrejahti-sovelluksen testaukseen käytetyt laitteet

Ominaisuus	iPhone 4	Samsung Galaxy S2	Nokia Lumia 820
Käyttöjärjestelmä	iOS 7	Android 4.1.2 Jelly Bean	Windows Phone 8
Näyttö (tuumaa)	3,5	4,27	4,3
Resoluutio (pikseliä)	960 x 640	800 x 480	800 x 480
Prosessori	Apple A4 800 MHz	ARM Cortex-A9 1,2 GHz tuplaydin	Snapdragon S4 1,5 GHz tuplaydin
RAM-muisti	512 Mb	1 Gt	1 Gt

Aarrejahti-sovelluksen koodia ei muunneltu eri kohdealustoja varten ollenkaan vaan sama koodi pakattiin kaikille kolmelle alustalle vain kerran. Koska Samsung Galaxy S2:ssa ja Nokia Lumia 820:ssa on sama näyttöresoluutio ja lähes samankokoinen näyttö, sovellus näytti näissä laitteissa lähestulkoon identtiseltä. Vain fontit näyttivät hieman eroavan toisistaan, mikä voidaan selittää eri selainmoottoreilla: Android-laitteissa on käytössä WebKit-selainmoottori kun taas Windows Phone -laitteet käyttävät Internet Explorer -mobiiliselainta.

iPhone 4:n näyttö on kooltaan testilaitteista pienin, mutta sen resoluutio on suurin. Sovellus ei mahtunut iPhonen näytölle aivan yhtä hyvin kuin Galaxy S2:n ja Lumia 820:n ja värit olivat haaleammat kuin muilla testilaitteilla (ks. kuvio 20, s. 54). iPhonella sovelluksen alanäyttöjen oikeaan yläreunaan sijoitettu takaisin-painike toimi vaihtelevasti: iOS-sovelluksille olisi voinut tehdä oman takaisin-painikkeen sovelluk-

sen alareunaan, jota WP8- ja Android-laitteilla ei näytettäisi, sillä näillä laitteilla on omat natiivit takaisin-painikkeensa. Lumia 820 ei kyennyt paikallistamaan käyttäjää yhtä nopeasti kuin iPhone tai Galaxy S2 vaan näytti välillä virheilmoituksen sijainnin määrityksen aikakatkaisusta: laite ei siis aina kyennyt paikallistamaan käyttäjän sijaintia sovellukseen määritellyssä vähimmäisajassa. Muutoin sovellus toimi hyvin kaikilla testilaitteilla ja erot eri laitteiden välillä olivat melko vähäisiä. Eri näkymien väliset siirtymät eli transitiot toimivat parhaiten Lumia 820:lla ja heikoiten Samsung Galaxy S2:lla.



Kuvio 20. Aarrejahti-sovellus Samsung Galaxy S2-, iPhone 4- ja Nokia Lumia 820 -älypuhelimissa

7.6 Jatkokehitys

Opinnäytetyötä varten tehdystä Aarrejahti-sovelluksesta jätettiin tarkoituksella kokonaan huomioimatta sovelluksen ylläpitopuoli, joka on yksi ensimmäisistä jatkokehityksessä tehtävistä asioista. Ylläpidon tulisi olla mahdollisimman vaivatonta ja helppoa, sillä kampanjoiden ylläpitäjät ovat usein tietämättömiä ohjelmoinnista ja ohjelmointikielistä. Aarrejahtien sijainnit tallennetaan JSON-tiedostoon leveys- ja pituusasteina: aarrejahdin ylläpitäjän täytyisi kyetä helposti tallentamaan uuden aarrejahdin sijainnin, kun aarteen kätköpaikka on selvillä. Tämän voisi mahdollisesti hoitaa joko omalla erillisellä sovelluksella, joka kehitettäisiin täysin Aarrejahdin ylläpitoa varten tai nykyiseen sovellukseen voitaisiin tehdä ylläpitäjien alue, jonne he voisivat kirjautua omilla tunnuksillaan. Ylläpitäjä voi aarrejulistetta tai -tarraa kiinnittäessään

tallentaa sen hetkisen sijaintinsa ylläpitosovelluksella aarteen sijainniksi. Samalla ylläpitäjä voisi ottaa mobiililaitteensa kameralla aarrejahdin molemmat vihjekuvat ja tallentaa ne aarteen sijainnin koordinaattien yhteydessä.

Toinen vaihtoehto ylläpitopuolen toteuttamiseen olisi tehdä verkkosivusto, jonne ylläpitäjät kirjautuisivat omien kampanjoidensa tunnuksilla. Sivustolla he voisivat helposti hallinnoida käynnissä olevia kampanjoita ja aarrejahteja: lisäillä uusia aarrejahteja kuvineen ja sijainteineen sekä muokata vanhoja. Aarrejahtien sijainnin tallentaminen voisi tapahtua kartan avulla: ylläpitäjä voisi merkitä karttaan paikan, jonne aarre on piilotettu. Karttaan merkityn sijainnin leveys- ja pituusasteet voitaisiin määrittää esimerkiksi Googlen geokoodaus-rajapinnan avulla. Sivusto voitaisiin myös yhdistää ylläpitosovelluksen kanssa, jolloin molempia voitaisiin käyttää aarrejahtien hallinnointiin.

Aarrejahti-sovelluksen menneille aarrejahdeille voitaisiin jatkokehityksessä toteuttaa omat näkymänsä, joissa vanhoja aarrejahteja pääsisi tarkastelemaan: menneistä aarrejahdeista voitaisiin kertoa muun muassa voittajan nimi, aarteen löytymisen ajankohta sekä paikka, josta aarre lopulta löytyi. Sovelluksen käyttöliittymää voitaisiin parantaa tekemällä iOS-alustoille oman takaisinpainikkeen, jota Android- tai Windows Phone 8 -laitteilla ei näytettäisi. Käyttöliittymäkomponenttien asettelua sekä kokoa voisi myös miettiä uudelleen niin, että sovellus mahtuu mobiililaitteen näytölle paremmin.

Aarrejahti-sovellukseen olisi hyvä saada mahdollisuus luoda oma aarteenetsijätili. Tällöin omat yhteystiedot voitaisiin tallentaa tunnukselle ja aarteen löydyttyä tarvitsisi täyttää vain aarteen koodi ja yhteystiedot voitaisiin hakea tunnuksen tiedoista. Omalla aarteenetsijätilillä voisi olla myös mahdollisuus säätää joitakin asetuksia, kuten aarteenetsinnän edistymiseen voitaisiin lisätä jokin ilmoitus: merkkiääni, värinä tai ponnahdusikkuna. Asetuksiin voisi myös määrittää sovelluksen värimaailman haluamakseen.

Aarrejahtien koodin voisi jatkokehityksessä korvata QR-koodilla, joka sitten luettaisiin sovelluksessa QR-koodilukijalla. Käyttäjälle voitaisiin myös ilmoittaa sovelluksen uu-

sista aarrejahdeista ja kampanjoista push-notifikaatiolla eli työntöilmoituksella, jolloin sovelluksen kuvakkeen yhteyteen tulee pieni ilmoitusikoni, kun uusia aarrejahteja tai kampanjoita on käynnistynyt. Tällä hetkellä sovellus toimii vain Jyväskylän alueella, mutta tulevaisuudessa se voitaisiin laajentaa eri kaupunkeihin: sovelluksen alussa voisi valita minkä kaupungin kampanjoihin tai aarrejahteihin haluaa osallistua tai halutut kaupungit voisi asettaa oman tilin tietoihin ja muokata niitä tarvittaessa. Sovellus voisi myös hakea lähimmät aarrejahdit käyttäjän sijainnin perusteella, jolloin ei tarvitsisi erikseen määritellä kaupunkia.

8 Pohdinta

Opinnäytetyössä tutkittiin alustariippumatonta hybridisovelluskehitystä mobiililaitteille ja siinä keskityttiin PhoneGap-sovelluskehykseen ja sen hyödyntämiseen hybridisovelluskehityksessä. Opinnäyteyössä selvitettiin PhoneGap-sovelluskehyksen soveltumista hybridisovelluskehitykseen sekä tutkittiin hybridisovellusten mahdollisuuksia, vahvuuksia ja heikkouksia verrattuna web- ja natiivisovelluksiin.

PhoneGap on tämän opinnäytetyön tekohetkellä yksi kaikkein suosituimpia ja käytetyimpiä hybridisovellusten toteuttamiseen kehitettyjä ohjelmistokehyksiä. PhoneGapin Apache-projektin, Apache Cordovan, ympärille on lyhyessä ajassa kehitetty monia käyttöliittymä- ja mobiilisovelluskehyksiä: muun muassa Ionic, AppGyver Steroids ja PhoneJS on suunniteltu käytettäväksi yhdessä nimenomaan Apache Cordovan kanssa. PhoneGapin ja Apache Cordovan korvaavia hybridisovelluskehyksiä ovat muun muassa Titanium Appcelerator ja Trigger.io, joista ensiksi mainitulla tehdyt sovellukset ovat PhoneGapilla tehtyjä sovelluksia lähempänä natiiveja: Appcelerator-sovellus kirjoitetaan puhtaasti JavaScriptillä, jonka Titaniumin rajapinta tulkkaa kunkin kohdealustan natiivikoodiksi. Trigger.io taas on toteutukseltaan Titaniumin Appceleratoria lähempänä PhoneGapia: siinä JavaScriptillä ja HTML:llä kirjoitettu sovellus toimii natiivin kehyksen sisällä ja laitteen natiiveihin ominaisuuksiin päästään käsiksi Trigger.io:n JavaScript-rajapinnan avulla.

PhoneGap-sovelluskehityksen yksi haasteellisimmista seikoista on sovelluksen testaus. PhoneGap-sovellusta voidaan kyllä web-sovelluksen tapaan testata internetselaimella, mutta selaimella ei pysty kokeilemaan natiivien ominaisuuksien toimivuutta. Opinnäytetyössä tehdyn Aarrejahti-sovelluksen sijaintirajapinnan tarkkuutta testattiin pääasiassa Ripple-emulaattorilla, joka soveltui testaukseen oikein hyvin. Ripplellä ei kuitenkaan saada testattua kaikkia PhoneGapin Javacript-rajapintoja: se tukee vain sijainti-, kiihtyvyysmittari- ja yhteysrajapintoja sekä joitakin PhoneGapin tapahtumia eli eventtejä (esimerkiksi deviceready). Testaaminen oikeilla laitteilla eroaa valitusta buildaus- eli paketointimenetelmästä: mikäli käytetään PhoneGap Build -pilvipalvelua, sovelluksen projektitiedostot tulee joka testauskerralla hakea uudelleen GitHubista, mikäli koodiin on tullut muutoksia, ja sovellus on pakattava uudel-

leen, jonka jälkeen se asennetaan testilaitteelle. Natiivia sovelluskehitystä muistuttavassa lähestymistavassa, jossa jokaisella eri kohdealustalla on oma kehitysympäristönsä, testaaminen mobiililaitteilla on nopeampaa: sovelluskehitysympäristöstä valitaan usb-kaapelilla kytketty laite sovelluksen testauskohteeksi, jolloin sovellus ajetaan paikallisesti laitteessa.

PhoneGap-sovelluskehityksen kaksi eri lähestymistapaa vaikuttavat myös suuresti sovelluskehitysprosessiin. Mikäli halutaan pakata sovellus ilman PhoneGap Build -palvelua, täytyy ensin valita sovelluksen pääkohdealusta, jonka sovelluskehitysympäristö asennetaan. Tämä lähestymistapa muistuttaa natiivisovelluskehitystä, sillä sovelluskehitystä varten täytyy ladata kohdealustan SDK ja sovellus kirjoitetaan ohjelmointityökalulla, kuten Eclipsellä Androidin tapauksessa tai Visual Studiolla Windows Phonelle. Web-kehittäjälle tämä lähestymistapa voi olla vieraampi, kun taas kokeneempi ohjelmoija valitsee varmimmin tämän lähestymistavan. Kuitenkin viimeistään siinä vaiheessa, kun sovellusta halutaan testata muillakin kuin alussa valitun pääkohdealustan laitteella, täytyykin olla muiden kohdealustoiden kehitysympäristöt, SDK:t ja työkalut asennettuna. Paras vaihtoehto olisikin tehdä jokaiselle eri kohdealustalle omat sovelluksensa erikseen omissa kehitysympäristöissään. Kerran kirjoitettu HTML, CSS ja JavaScript voidaan kuitenkin käyttää lähestulkoon kokonaan uudelleen kohdealustasta riippumatta.

PhoneGap Build -lähestymistapa muistuttaa taas hyvin pitkälti web-sovelluskehitystä: siinä käytettävällä tietokoneella ei ole niinkään merkitystä ja sovelluskehittäjä saa itse valita mieluisensa koodieditorin sovelluksensa kirjoittamiseen. Testaaminen mobiililaitteilla on kuitenkin hankalampaa kuin natiivimmassa lähestymistavassa sekä PhponeGap Buildin tuki erilaisille liitännäisille eli plugineille ei ole täydellinen. PhoneGap Buildilla pakattun sovelluksen testaaminen sekä Android-, iOS- että Windows Phone 8 -laitteilla vaatii vähintään Windows 8 -käyttöjärjestelmäisen tietokoneen: Android-sovelluksia voidaan testata tietokoneen käyttöjärjestelmästä huolimatta, iOS-sovellukset vaativat iOS-allekirjoitusavaimen muuntamista P12-tiedostoksi, joka onnistuu niin MAC- kuin Windows-laitteilla. Windows Phone 8 taas vaatii Windows 8 -käyttöjärjestelmän testilaitteen rekisteröimiseksi sovelluskehitystä varten sekä sovelluskaupan ulkopuolisen sovelluksen asentamiseksi tähän laitteeseen.

Tämän opinnäytetyön tekohetkellä kaikkein vaivattomin tapa sovelluskaupoissa jaeltavien alustariippumattomien sovellusten tekoon lienee Aarrejahti-sovelluksessa käytetty PhoneGap Build -pilvipalvelu. PhoneGap Build ei tosin sovellu joka tilanteeseen eikä sillä voida korvata natiiveja sovelluksia: parhaiten se sopii yksinkertaisten web-sovellusten paketointiin natiiveiksi. PhoneGapin natiivia muistuttava lähestymistapa mobiilisovelluskehitykseen mahdollistaa jo monimutkaisempien sovellusten toteutuksen ja siinä voidaan huomioida kukin eri kohdealusta erikseen, mikä taas parantaa käyttäjäkokemusta. Sovelluskehittäjällä täytyy tosin olla tällöin kohdealustojen vaatimat kehitysympäristöt ja laitteet käytössään, mutta hän pystyy kierrättämään melko vaivattomasti kerran kirjoitettua koodia eri sovellusten välillä.

Hybridisovellus on hyvä vaihtoehto web-kehittäjille ja eri kohdealustojen vaatimiin ohjelmointikieliin perehtymättömille. Hybridisovellusten kehitys on usein nopeampaa sekä huokeampaa kuin natiivisovellusten, mutta etuina web-sovelluksiin nähden ovat niiden jakelu sovelluskaupoissa ja joidenkin mobiililaitteen natiivien ominaisuuksien hyödyntäminen. Hybridisovellusten toteuttamiseen tarkoitetut työkalut kehittyvät jatkuvasti ja uusia ollaan koko ajan kehittämässä. Hybridisovellusten tarjoamat mahdollisuudet kiinnostavat sovelluskehittäjiä; ICT-tutkimuksen kärkeä edustava Gartner on arvioinut hybridisovellusten määrän olevan kasvussa ja että vuoteen 2016 mennessä jopa puolet kaikista mobiilisovelluksista olisi hybridejä.

Lähteet

Android Pushes Past 80% Market Share While Windows Phone Shipments Leap 156.0% Year Over Year in the Third Quarter, According to IDC. 2013. Viitattu 19.02.2014. http://www.idc.com/getdoc.jsp?containerId=prUS24442013

Apple Developer Programs. 2014. Viitattu 07.04.2014. https://developer.apple.com/programs/

Arki muuttuu yhä mobiilikeskeisemmäksi. 2013. Viitattu 18.02.2014. http://www.tns-gallup.fi/uutiset.php?aid=14935&k=14320

Avola, G. & Raasch, J. 2013. Smashing *Mobile Web Development*. Yhdysvallat: Wiley.

Boonstra, L. 2013. *Getting Started with Sencha Touch 2: Build a Weather Utility App (Part 3).* Viitattu 07.04.2014. http://www.sencha.com/blog/getting-started-with-sencha-touch-2-build-a-weather-utility-app-part-3

Budiu, R. 2013. *Mobile: Native Apps, Web Apps, and Hybrid Apps.* Viitattu 24.02.2014. http://www.nngroup.com/articles/mobile-native-apps/

Colom, A., Magnuson M. & Sotag, A. 2014. *Getting Started with jQuery Mobile.* Viitattu 13.04.2014. https://learn.jquery.com/jquery-mobile/getting-started/

Davis, L. 2009. *PhoneGap: People's Choice Winner at Web 2.0 Expo Launch Pad.* Viitattu 10.03.2014. http://readwrite.com/2009/04/02/phone_gap_todays_peoples_choice_winner_at_launch_p#awesm=~oy7Qd52QS9klT2

Frommel, O. 2013. *Developing an App for iOS, Android and Windows Phone - a Comparative Study.* Viitattu 24.02.2014. http://www.zetalab.de/blog/developing-an-app-for-ios-android-and-windows-phone-a-comparative-study/

Gartner Says by 2016, More Than 50 Percent of Mobile Apps Deployed Will be Hybrid. 2013. Viitattu 24.03.2014. http://www.gartner.com/newsroom/id/2324917

Gerbasi, R. 2013. *Leveraging PhoneGap within Sencha Touch.* Viitattu 07.04.2014. http://phonegap.com/blog/2013/11/20/SenchaPhoneGap/

Ghatol, R. & Patel, Y. 2012. *Beginning PhoneGap: Mobile Web Framework for JavaScript and HTML5.* New York: Apress.

Hynninen, T. 2013. *Androidin historia: Astrosta Jelly Beaniin.* Viitattu 18.02.2014. http://www.mobiiliblogi.com/2013/07/20/androidin-historia-astrosta-jelly-beaniin/

iOS: A visual history. 16.09.2013. Viitattu 19.02.2013. http://www.theverge.com/2011/12/13/2612736/ios-history-iphone-ipad

jQuery. 2014. *jQueryn virallinen verkkosivusto.* Viitattu 13.04.2014. http://jquery.com/

jQuery Mobile 1.4 Supported Platforms. 2014. Viitattu 13.04.2014. http://jquerymobile.com/gbs/1.4/

Kodintekniikkaindeksi. 2013. Viitattu 18.02.2014. http://mb.cision.com/Public/1648/9491378/a94797fc18bf22f5.pdf

Koskimies, K. & Mikkonen, T. 2005. *Ohjelmistoarkkitehtuurit.* Jyväskylä: Talentum Media Oy.

Kosmaczewski, A. 2013. *Sencha Touch 2 Up and Running.* Yhdysvallat: O'Reilly Media Inc.

Leroux, B. 2012. *PhoneGap, Cordova, and what's in a name?* Viitattu 10.03.2014. http://phonegap.com/2012/03/19/phonegap-cordova-and-what%E2%80%99s-in-a-name/

Moore, D. 2013. *Developing Cross Platform Mobile Applications with Cordova CLI.* Viitattu 05.04.2014. https://leanpub.com/developingwithcordovacli/read

One In Every 5 People In The World Own A Smartphone, One In Every 17 Own A Tablet. 2013. Viitattu 18.02.2014. http://www.businessinsider.com/smartphone-and-tablet-penetration-2013-10

Ortiz, C. 2011. *Introduction to jQuery Mobile*. Viitattu 13.04.2014. http://www.ibm.com/developerworks/library/wa-jqmobile/

PhoneGap. 2014. *PhoneGapin virallinen verkkosivusto.* Viitattu 10.03.2014. http://phonegap.com/

PhoneGap Build Documentation. 2014. Viitattu 05.04.2014. http://docs.build.phonegap.com/

PhoneGap Documentation. 2014. Viitattu 10.03.2014. http://docs.phonegap.com/en/3.4.0/index.html

Platform Versions. 2014. Viitattu 24.02.2014. http://developer.android.com/about/dashboards/index.html#Platform

Ruby Programming Language. 2014. Viitattu 02.04.2014. https://www.ruby-lang.org/en/

Sencha Docs. 2014. Viitattu 02.04.2014. http://docs.sencha.com/touch/2.3.1/

Sencha Touch Build Mobile Web Apps with HTML5. 2014. Viitattu 28.03.2014. http://www.sencha.com/products/touch/features/

Trice, A. 2012. *PhoneGap Explained Visually.* Viitattu 09.03.2014. http://phonegap.com/2012/05/02/phonegap-explained-visually/

Riippi, J. 2013. *Natiivi, hybridi ja HTML5.* Viitattu 02.03.2014. http://67.prosenttia.fi/2013/05/27/natiivi-hybridi-ja-html5/

Taft, D. 2009. *PhoneGap Simplifies iPhone, Android, BlackBerry Development.* Viitattu 10.03.2014. http://www.eweek.com/c/a/Application-Development/PhoneGap-Simplifies-iPhone-Android-BlackBerry-Development-788189/

Traeg, P. 2014. *Four Ways To Build A Mobile Application, Part 3: PhoneGap.* Viitattu 14.04.2014. http://www.smashingmagazine.com/2014/02/11/four-ways-to-build-a-mobile-app-part3-phonegap/

Usage of JavaScript libraries for websites. 2014. Viitattu 13.04.2014. http://w3techs.com/technologies/overview/javascript_library/all

Vino, V. 2014. *Introduction to M-V-C in sencha touch 2.3.* Viitattu 29.03.2014. http://techwaka.com/mobile/introduction-mvc-sencha-touch

Windows Phone noussut Suomen johtavaksi älypuhelinalustaksi. 2013. Viitattu 24.02.2014. http://www.marketvisio.fi/fi/ajankohtaista/uutiset-marketvisio/1703-windows-phone-noussut-suomen-johtavaksi-lypuhelinalustaksi